Titre: Title:	Optimisation paramétrique de circuits analogiques par le biais des algorithmes génétiques
Auteur: Author:	Hugues Langlois
Date:	2003
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Langlois, H. (2003). Optimisation paramétrique de circuits analogiques par le biais des algorithmes génétiques [Master's thesis, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/7132/

In compliance with the
Canadian Privacy Legislation
some supporting forms
may have been removed from
this dissertation.

While these forms may be included
in the document page count,
their removal does not represent
any loss of content from the dissertation.

UNIVERSITÉ DE MONTRÉAL

OPTIMISATION PARAMÉTRIQUE DE CIRCUITS ANALOGIQUES PAR LE BIAIS DES ALGORITHMES GÉNÉTIQUES

HUGUES LANGLOIS

DÉPARTEMENT DE GÉNIE ÉLECTRIQUE

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉE EN VUE DE L'OBTENTION

DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES (M.SC.A.)

(GÉNIE ÉLECTRIQUE)

JUILLET 2003

Your file Votre référence
ISBN: 0-612-86408-1
Our file Notre référence
ISBN: 0-612-86408-1

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

CE MÉMOIRE INTITULÉ :

OPTIMISATION PARAMÉTRIQUE DE CIRCUITS ANALOGIQUES PAR LE BIAIS DES ALGORITHMES GÉNÉTIQUES

présenté par: Hugues Langlois

en vue de l'obtention du diplôme de: Maîtrise ès sciences appliquées (M.Sc.A.)

a été dûment accepté par le jury d'examen constitué de:

M. AUDET Yves, Ph. D., président-rapporteur

M. SAVARIA Yvon, Ph. D., membre et directeur de recherche

M. SAWAN Mohamad, Ph. D., membre du jury

À mes parents

« J'ai échoué là où je n'ai pas réussi »
Savoir continuer là où l'échec est probant.
Toujours relever le défi.

Remerciements

Je tiens à remercier mon directeur de recherche Yvon Savaria pour m'avoir permis de découvrir et de m'investir dans le domaine de la microélectronique analogique.

Je tiens à remercier LTRIM Technologies Inc., et plus particulièrement Yves Gagnon, pour son soutien financier et pour avoir participé à la concrétisation de cette idée, lorsqu'elle était encore à son stade embryonnaire.

Je tiens à remercier Bertrand, mon père, pour m'avoir sensibilisé à l'existence des algorithmes génétiques dès mes premières années de mon baccalauréat en génie.

Je me dois de remercier Danyèle, ma mère, pour avoir toujours désiré pour moi le meilleur de ce monde et pour avoir été de tout cœur avec moi à tous les instants.

Je remercie Annie Poirier pour son merveilleux soutien moral et affectif tout au long de ce long processus qu'est la maîtrise.

Je remercie Jean-Marc Tremblay pour m'avoir incité à poursuivre mes études aux cycles supérieurs.

Je me dois de remercier Abdelhalim Bendali pour avoir été, sans le savoir, un élément décisif et précurseur à mon implication dans le domaine de l'optimisation des circuits analogiques.

Je tiens particulièrement à remercier Marc Bertola pour son soutien et son aide en programmation sur Builder C++, lors de la réalisation de ma première version du logiciel.

Je tiens à remercier Simon Rioux pour toutes les discussions constructives et soutenues à propos de l'opération du logiciel d'optimisation.

Je tiens encore plus particulièrement à remercier David Marche pour son travail exceptionnel de programmeur lors de l'implémentation de la version améliorée du logiciel.

Je tiens à remercier Adel Belhaouane et Chokri Achour pour leurs excellents travaux au niveau des dessins des masques qui ont permis de valider le fonctionnement des circuits synthétisés.

Je tiens à saluer Jean-François Delage, Mathieu Renaud, Marius Tizu et Abdelaziz Trablesi, pour leurs rôles dans l'évaluation du logiciel d'optimisation.

Le logiciel utilisé dans ce travail fonctionne à l'aide de la bibliothèque : *Galib genetic algorithm (GALib)*, réalisée par Matthew Wall à l'Institut des Technologies du Massachusetts (MIT).

Résumé

La synthèse de circuits analogiques peut être traduite comme une tâche d'optimisation d'un problème multi-objectif. On associe normalement aux différents objectifs des performances fonctionnelles, des spécifications électriques ou des contraintes technologiques.

Un logiciel d'optimisation de circuits analogiques a été développé. Ce logiciel conjugue l'utilisation d'un simulateur de circuit pour l'évaluation des performances et d'un algorithme heuristique pour réaliser la tâche d'optimisation. L'algorithme d'optimisation employé pour résoudre ce problème multi-objectif est basé sur les algorithmes génétiques. L'algorithme génétique est une technique d'optimisation informatique qui emprunte plusieurs opérateurs et principes sous-jacents à l'évolution biologique.

Une approche sur une base graphique et interactive a été intégrée à ce logiciel afin de permettre à l'usager de facilement formuler ses préférences au cours de la progression de l'optimisation du circuit. Cet outil se classe dans une philosophie de résolution de la *programmation par but*. Ce qui signifie que chaque fonction de coût se fait par rapport à un vecteur de référence défini par l'usager. Le but de ce mémoire est de présenter un outil d'optimisation paramétrique de circuits analogiques qui automatise l'étape de dimensionnement des composantes dans le flot de conception traditionnel.

Abstract

The synthesis of analog circuits is a form of multi-objective optimization problem. One associates the various objectives to desired functionalities or performances, related to electrical specifications and restrictions as well as technological constraints.

An optimization software for analog circuits was developed. This software combines the use of a circuit simulator with a heuristic algorithm. The optimization algorithm employed to solve this multi-objective problem is based on genetic algorithms. Genetic algorithms are data-processing optimization techniques of that borrow several principles and operators to biological evolution processes.

An interactive approach supported by a graphic interface was implemented into a software that supports decision makers in the process of easily formulating preferences while progressing through the process of optimizing an analog circuit. This method can be classified as a goal programming approach. This means that each performance index is compared to a reference vector defined by the decision maker.

The goal of this master is to present a parametric optimization tool for analog circuits that automates the process of sizing components in the traditional design flow. Using such a tool, it is now possible to improve the process of designing high performance analog circuits.

Table des matières

Liste des figures

Liste des tableaux

Table des annexes

Liste des listages

Définitions

Analyse DC	Analyse dans un domaine paramétrique pouvant être une tension, un courant, la température, etc.
Analyse AC	Analyse dans le domaine fréquentiel
Analyse TR	Analyse dans le domaine temporel
Analyse Monte Carlo	Analyse par perturbation aléatoire révélant statistiquement les probabilités de bon fonctionnement d'un circuit
Candidat Monte Carlo	Concerne *un* des multiples essais aléatoires de l'analyse Monte Carlo
Décideur	Coordonnateur de l'optimisation
Fonction de pointage	Transformation d'une performance en valeur numérique
Fonction d'utilité	Combinaison mathématique de plusieurs fonctions de pointage
Fonction de désirabilité	Idem à fonction de pointage
Fonction de coût	Idem à fonction de pointage
Multi-courbes	Courbe de performance définie selon une première variables indépendante continue et une seconde variable indépendante de type discrète (aussi appelé multi-sweep ou tables de courbes)
Netlist	Description textuelle d'un circuit analogique dans un format compatible avec les logiciels de simulation électronique
Solution supportée	Solution retrouvée à la frontière Pareto Optimales (voir Figure 1.4)
Solution non-supportée	Solution optimale mais située sur une frontière concave (voir Figure 1.1)

Solution dominée	Solution à l'intérieur de l'enveloppe des solutions réalisables
Solution non-dominée	Solution Optimale au sens Pareto
Solveur	Programme qui réalise l'optimisation
Plage de T° commerciale	0 °C à 70 °C
Plage de T° industrielle	–40 °C à 85 °C
Plage de T° militaire	–55 °C à 125 °C

Liste des abbréviations

AC	Analyse dans le domaine fréquentiel
AG	Algorithme Génétique
AGO	Analog Genetic Optimizer
AHDL	Analog Hardware Description Language
ASCII	American Standard Code for Information Interchange
ASIC	Application-Specific Integrated Circuit
C	Préfixe pour les capacités
CA	Circuit Analogique
CAO	Conception Assistée par Ordinateur
CPOLY	Capacité faite en polysilicium
CPU	Central Processing Unit
dB	décibel
DC	Analyse dans un domaine paramétrique (V,I,T°,...) ou du point d'opération
d.d.p.	différence de potentiel
ex	exemple
GA	Genetic Algorithms
GND	Masse (Ground)
GP	Genetic Programming
GP	Geometric Programming
GP	Goal Programming
GRM	Groupe de Recherche en Microélectronique
IEEE	Institute of Electrical and Electronics Engineers
LM	Levenberg-Marquardt
M	Préfixe pour les transistors MOSFET
MC	Monte Carlo
MIT	Massachusetts Institute of Technology

N	Préfixe pour les noms de nœuds
PC	Personal Computer
PLL	Phase-Lock Loop
PMO	Problème Multi-Objectif
PO	Pareto Optimal
PSR	Power Supply Rejection
PSRR	Power Supply Ratio Rejection
PTAT	Proportional To Absolute Temperature
Q	Préfixe pour les transistors bipolaires
R	Préfixe pour les résistances
RPOLY2	Résistance de polysilicium ($2^{ème}$ couche)
RT	Recherche Tabou
SPICE	Simulation Program with Integrated Circuit Emphasis
TC	Temperature Coefficient
TEMPCO	Temperature Coefficient
TR	Analyse dans le domaine temporel
TS	Tabu Search
TSMC	Taiwan Semiconductor Microelectronics Company
UNIX	Système d'exploitation
VBG	Tension de Bandgap (~1.23V)
VCL	Virtual Components Library
VDD	Alimentation positive (+3.3V)
VHDL	VHSIC Hardware Description Language
VHDL-A	VHSIC Hardware Description Language for Analog
VHSIC	Very High Speed Integrated Circuit
WYSIWYG	What You See Is What You Get

Introduction

Un grand nombre de publications traitent d'outils d'aide à la Conception Assistée par Ordinateur (CAO). Les dernières décennies ont surtout été marquées par la réalisation d'outils CAO très performants du côté numérique de la microélectronique. Cette tendance a surtout été stimulée par l'arrivée des technologies de télécommunication dans la vie de tous les jours. Cet engouement pour l'automatisation de la synthèse ne s'est fait ressentir que bien plus tard du côté de la microélectronique analogique. Ce retard est associé à la nature plus complexe du problème adressé par la synthèse analogique.

En général, les outils de synthèse analogique suivent le modèle présenté à la Figure 0.1.

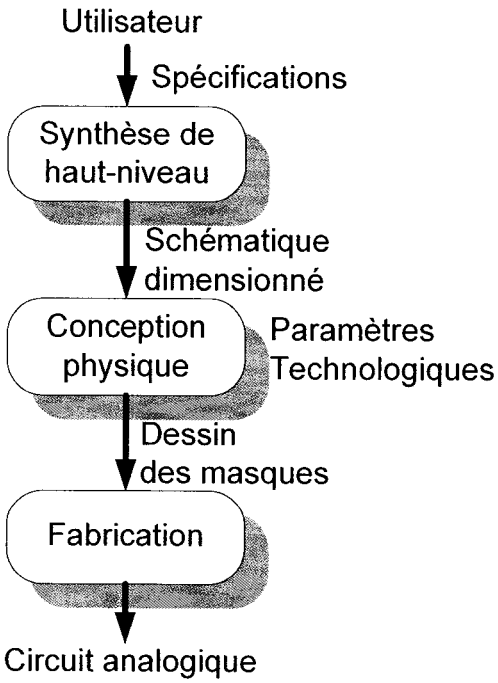


Figure 0.1 Le processus de synthèse analogique.

Les premiers outils de *simulation analogique* ont vu le jour presque en même temps que les outils de *simulation numérique*. Mais la synthèse numérique s'est plus rapidement développée grâce aux langages de programmation comme le VHDL (VHSIC (Very High Speed Integrated Circuit) Hardware Description Language)[82] ou le VeriLog permettant au concepteur, par le fait même, de se propulser à des niveaux d'abstraction plus élevés.

Durant cette période de grands progrès du côté numérique, la méthodologie de conception des circuits analogiques (CA) continuait majoritairement à se faire manuellement. Bien que les logiciels de simulation et les modèles des composants électroniques ont grandement évolué au fil des années, le simulateur de circuit reste un outil de vérification permettant de connaître, durant la phase de conception, les performances attendues de ses circuits. Notons bien que cette boucle de conception se résume généralement à une fastidieuse tâche itérative d'optimisation basée sur l'essai et l'erreur.

Encore aujourd'hui des tentatives d'utiliser des langages analogiques comme le AHDL ou le VHDL-A restent très limitées et ne font pas véritablement l'unanimité dans la communauté. Le problème réside essentiellement dans le fait que la conception d'un CA est une tâche extrêmement complexe qui fait intervenir plusieurs étapes qu'il est difficile de substituer ou d'automatiser par des algorithmes.

La Figure 0.2 représente la structure organisationnelle du processus de conception analogique. Les nombres associés aux étapes sont expliqués dans le Tableau 0.1.

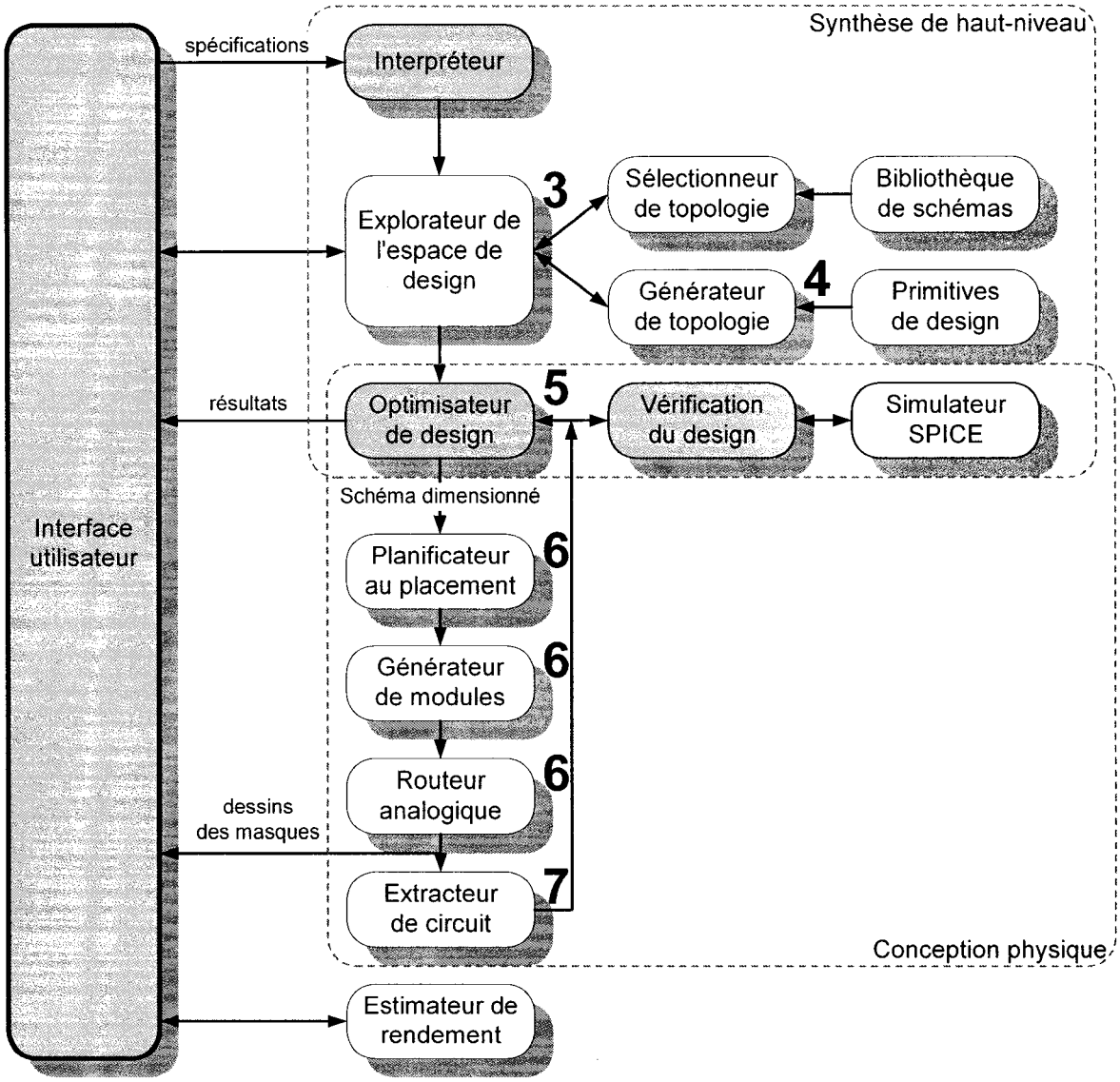


Figure 0.2 Structure organisationnelle de la synthèse analogique.

Le Tableau 0.1 présente le degré de complexité et l'expertise sollicitée, pour chacune des étapes critiques à effectuer, lors de la réalisation d'un CA. Les zones ombragées représentent les tâches normalement effectuées par le concepteur. L'automatisation de l'étape 5 sera le sujet de ce mémoire.

Tableau 0.1 Étapes du processus de conception d'un circuit analogique.

	Marche à suivre (Étapes)	Expertise sollicitée	Complexité associée
1.	Se restreindre aux composantes disponibles dans le procédé	limitation des composantes	simple
2.	S'assurer de la fiabilité de la modélisation des composantes	validation des modèles	simple
3.	Sélectionner ou créer une topologie de circuit	concepteur	très complexe
4.	Déterminer les contraintes structurelles	concepteur	complexe
5.	Dimensionner les composants	concepteur	assez complexe
6.	Placer et interconnecter	concepteur	difficile
7.	Vérifier l'impact du placement et routage sur les spécifications du CA (extraction)	concepteur	systématique
8.	Réalisation du CA	fabrication	systématique
9.	Mesurer les performances réelles du CA et réitérer à l'étape 5	validation expérimentale	de simple à difficile

La sélection d'une topologie de CA (étape 3) est une tâche très complexe, puisque le *domaine de recherche* qui lui est associé est de type discontinu et qu'une sélection arrêtée s'accompagne généralement par des limitations fondamentales sur les performances. Étant donné que l'étape du choix topologique est importante et qu'elle constitue un prélude à la phase d'optimisation paramétrique (étape 5), il peut paraître intéressant d'intégrer celle-ci directement dans le *processus de recherche*. Diverses approches ont été investigués dans cette voie, mais on distingue plus particulièrement : les algorithmes génétiques (AG) [33][62], la programmation génétique [50], la programmation géométrique [39][63], et le recuit simulé multi-moléculaire [105]. Pour la plupart des approches d'optimisation intégrant des modifications topologiques, l'espace de recherche se veut limité à des variantes topologiques voisines et envisagées par le concepteur. Seule la *programmation génétique* offre à la recherche de CA, une liberté complète sur la sélection de la topologie du circuit (section 1.2.2.3).

Dans ce mémoire, l'exploration des variantes topologiques ne sera pas priorisée, bien que celles-ci peuvent être intégrées selon la description de circuit relatée dans [105] par l'ajout de variables booléenne additionnelles pouvant activer ou de désactiver des composantes du circuit. De plus, le placement et routage [49] des transistors (étapes 6 et 7) ne seront pas non plus abordées, car au stade de conception où nous situons notre recherche, il n'est pas prioritaire de tenir compte des impacts apportés par l'extraction des éléments parasites apparaîssant lors de l'implémentation physique.

Ce mémoire portera principalement sur l'étape du dimensionnement paramétrique (étape 5) des composantes électroniques du CA. Nous assumerons que le concepteur saura intégrer à une topologie de circuit de son choix, plusieurs connaissances *a priori* comme les *contraintes structurelles* ou *appariements* entre les transistors ainsi que la *plage paramétrique* allouée à chacun des composants électroniques du CA. Par l'ajout de ces connaissances *a priori*, la dimensionnalité de l'espace de recherche peut être ainsi significativement réduite, ce qui a pour incidence de favoriser la convergence. De plus, l'apport de connaissances *a priori* a aussi pour effet d'exclure de l'investigation les circuits présentant de fortes probabilités de mauvaise fonctionnalité [107]. Un exemple typique de connaissance *a priori* concerne l'appariement des tailles d'une paire différentielle ainsi que les tailles d'un miroir de courant (section 3.1.4).

Les outils existants de synthèse analogique ont toujours tenté de trouver le meilleur compromis possible entre la *couverture* de l'espace des solutions, la *qualité* de l'évaluation des circuits et la *vitesse* de convergence de l'optimisation. La stratégie adoptée dans ce mémoire réalise une recherche de type *optimum global*. La qualité de l'évaluation est assurée par HSpice. Enfin, la durée de convergence demeure cependant variable à cause de la *nature heuristique* du processus d'optimisation.

Dans le chapitre 1, nous présenterons la revue de la littérature sur l'optimisation de circuits analogiques. Le chapitre 2 portera sur la présentation du logiciel d'optimisation AGO et ses détails internes. Ensuite, le chapitre 3 présentera une

configuration typique pour l'optimisation d'un circuit de référence de tension. Enfin le chapitre 4 conclura ce mémoire.

1 Revue de littérature

Cette section trace un portrait général des types de problèmes et de solutions auxquels se réfèrent l'optimisation des CA. De plus, les différents algorithmes de résolution tirés de la littérature et appliqués de manière pratique à l'optimisation de CA seront détaillés. Enfin, un tableau résumé des outils existants (académiques et commerciaux) qui réalisent l'optimisation de CA conclura cette revue de littérature.

1.1 L'optimisation de circuits analogiques : un problème multi-objectif

La synthèse automatisée des CA se généralise comme un problème multicritères[1] et de grande dimension pour lequel des paramètres doivent être déterminés de manière optimale. Il y a généralement plusieurs performances contradictoires à satisfaire simultanément ainsi que plusieurs contraintes à respecter sur les paramètres optimisables. L'*optimisation multi-objectif* est la spécialité qui s'intéresse à la résolution de ce type de problèmes. Cette discipline tire ses racines de la fin du 19ème siècle d'après les travaux en économie de Edgeworth et Pareto [1].

L'optimisation multi-objectif cherche donc à optimiser plusieurs composantes d'un vecteur de fonction de coût. Contrairement à l'optimisation uni-objectif, la solution d'un *problème multi-objectif* (PMO) n'est pas une solution unique, mais un ensemble de

[1] On utilise de manière indifférente les termes objectif, performance, critère ou attribut.

solutions, connu comme l'ensemble des solutions *Pareto Optimales* (PO)[2]. Toute solution de cet ensemble est optimale, c'est-à-dire qu'*aucune amélioration* ne peut être faite sur une composante du vecteur *sans dégradation* d'au moins une autre composante du vecteur.

Le premier objectif dans la résolution d'un problème multi-objectif est d'obtenir l'ensemble PO des solutions Pareto optimales ou d'échantillonner le plus uniformement possible des solutions à partir de l'ensemble PO. Normalement, la détermination de l'ensemble PO n'est qu'une première phase dans la résolution pratique de PMO, qui nécessite dans un deuxième temps le choix d'une solution à partir de cet ensemble suivant des préférences choisies par le décideur[3]. Le choix d'une solution par rapport à une autre nécessite la connaissance du problème et des nombreux facteurs liés à celui-ci. Ainsi, une solution retenue par un décideur peut ne pas être acceptable pour un autre décideur. Il est donc utile d'avoir plusieurs alternatives dans le choix d'une solution Pareto optimale.

La difficulté dans la conception d'un solveur de PMO réside dans les faits suivants :

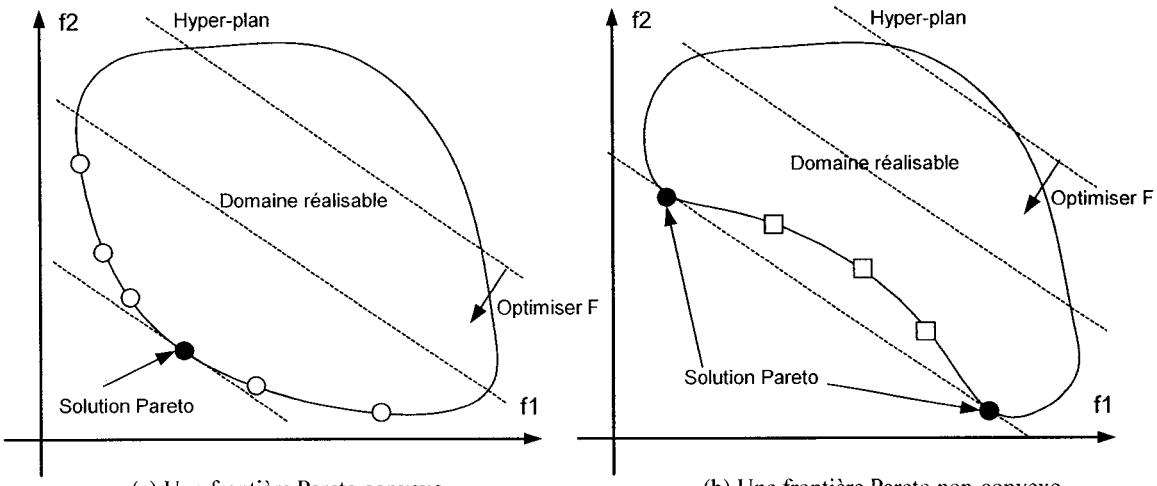
- Il n'y a pas de définition communément admise sur l'optimalité d'une solution comme en optimisation uni-objectif. La relation d'ordre entre les solutions du problème n'est que partielle, et le choix final revient au décideur.

---

[2] Cet ensemble est aussi appelé frontière Pareto.

[3] Le décideur est l'individu qui encadre l'optimisation.

- Le nombre de solutions optimales (au sens de Pareto) augmente en fonction de la taille du problème et principalement avec le nombre de critères utilisés.
- Pour les PMO convexes (Figure 1.1a), on trouve *une seule* solution Pareto Optimale située à la frontière des solutions réalisable pour chaque formulation de l'optimisation.
- Pour les PMO non convexes (Figure 1.1b), les solutions Pareto sont localisées dans les frontières des solutions réalisables, tangentes à l'hyperplan d'optimisation et à l'intérieur de l'enveloppe convexe.



(a) Une frontière Pareto convexe (b) Une frontière Pareto non-convexe

Figure 1.1 Solutions Pareto Optimales privilégiées par la méthode d'agrégation selon la convexité de l'espace de recherche pour un PMO simplifié à deux contraintes (f1, f2) : (a) PMO convexe (un seule solutions optimale, non-dominée), (b) PMO non convexe (plusieurs solutions optimales).

Une des questions fondamentales dans la résolution de PMO concerne la coopération entre le solveur et le décideur. Cette coopération peut prendre une des trois formes suivantes [98]:

- **A priori** : les solutions proposées pour résoudre les PMO consistent souvent à combiner les différentes fonctions coût suivant une certaine *fonction d'utilité*, pour obtenir une seule fonction à optimiser. Dans ce cas le décideur est supposé connaître *a priori* le poids de chaque objectif ainsi que la fonction d'utilité. Ceci revient donc à transformer un PMO en un problème uni-objectif et à le résoudre par des méthodes d'optimisation classique. Cependant, dans la plupart des cas, la fonction d'utilité n'est pas connue *a priori* du processus d'optimisation, et les différents objectifs sont parfois non-commensurables[4]. De plus, l'espace de recherche défini peut ne pas représenter effectivement le problème initial. Si le décideur n'est pas à même d'indiquer *a priori* le type de compromis qu'il souhaite réaliser entre les critères, il n'est pas pertinent de chercher une et une seule solution efficace réalisant une agrégation entre ces critères. Outre la combinaison des différents critères présents dans l'optimisation, le type de connaissances *a priori* normalement intégrées avec les CA concerne les contraintes structurelles et les plages paramétriques des composantes.
- **A posteriori**: le décideur choisit une solution parmi les solutions de l'ensemble PO fourni par le solveur. Cette approche est utilisable dans le cas où la cardinalité[5] de l'ensemble PO est réduite. Dans le cas contraire, pour l'aider à prendre une décision, il convient de lui permettre d'explorer l'ensemble des solutions en

---

[4] Des objectifs sont non-commensurables si leurs valeurs sont exprimées dans des unités différentes. Par exemple, si on cherche, pour un circuit analogique, à minimiser la sensibilité aux fluctuations de l'alimentation (dB) par rapport à la consommation en courant (μA).

[5] La cardinalité est la quantité d'éléments présents dans un ensemble.

fonction de ses préférences, afin qu'il puisse mieux appréhender les arbitrages à opérer entre les performances.

- **Interactive**: dans ce cas, il y a coopération progressive entre le décideur et le solveur (Figure 1.2). À partir des connaissances acquises pendant la résolution du problème, le décideur définit des préférences. Ces préférences sont prises en compte par le solveur dans la résolution du problème. Ce processus est réitéré pendant plusieurs étapes. À l'issue de l'exploration guidée de l'ensemble PO, le décideur dispose d'une connaissance approfondie pour retenir une solution de l'ensemble PO représentant un compromis acceptable. Un grand spécialiste [87] du domaine de la programmation multi-objectif considère même l'approche interactive comme étant "probablement l'approche la plus prometteuse pour le domaine de l'optimisation multi-objectif".

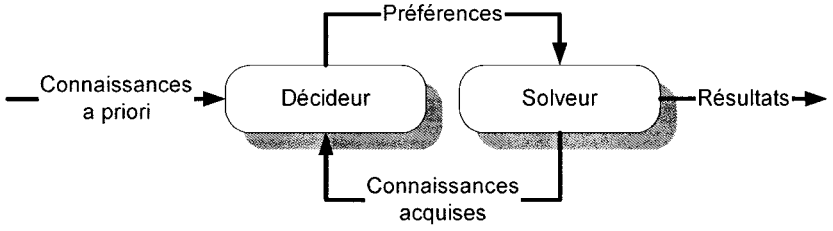


Figure 1.2 Approche interactive: coopération entre le solveur et le décideur.

Avant de présenter en détail les algorithmes de résolution, une classification des différentes méthodes d'optimisation disponibles pour la résolution des PMO sera présentée. Cet étalage global des approches de résolution permettra au lecteur de mieux appréhender le contexte et le positionnement de ce mémoire.

1.1.1 Classification des méthodes d'optimisation multi-objectif

Des méthodes sont nécessaires pour résoudre les problèmes de grandes tailles et/ou les problèmes avec un nombre de critères supérieurs à deux. Elles ne garantissent pas de trouver de manière exacte l'ensemble PO, mais une approximation de cet ensemble noté PO^*. Les méthodes heuristiques peuvent être divisées en deux classes: d'une part les algorithmes spécifiques à un problème donné qui utilisent des connaissances du domaine et d'autre part les algorithmes généraux applicables à une grande variété de PMO. Notre intérêt dans ce mémoire porte sur la deuxième classe d'algorithmes, les *métaheuristiques*. Plusieurs adaptations de métaheuristiques ont été proposées dans la littérature pour la résolution de PMO et la détermination des solutions Pareto : le recuit simulé, la recherche tabou, les algorithmes évolutionnaires et les algorithmes génétiques.

Les approches utilisées pour la résolution de PMO peuvent être classées en trois catégories (Figure 1.3) :

- **Approches basées sur la transformation du problème en un problème uni-objectif** : Cette classe d'approche comprend par exemple les méthodes basées sur l'agrégation qui combinent les différentes fonctions coût f_i du problème en une seule fonction objectif F (section 1.1.2.1). Ces approches nécessitent, de la part du décideur, d'avoir une bonne connaissance de son problème.

- **Approches non-Pareto** : Les approches non-Pareto ne transforment pas le PMO en un problème uni-objectif. Elles utilisent des opérateurs de recherche qui traitent séparément les différents objectifs.
- **Approches Pareto** : Les approches Pareto utilisent directement la notion d'optimalité Pareto [106] dans leur processus de recherche. Le processus de sélection des solutions générées est basé sur la notion de non-dominance.

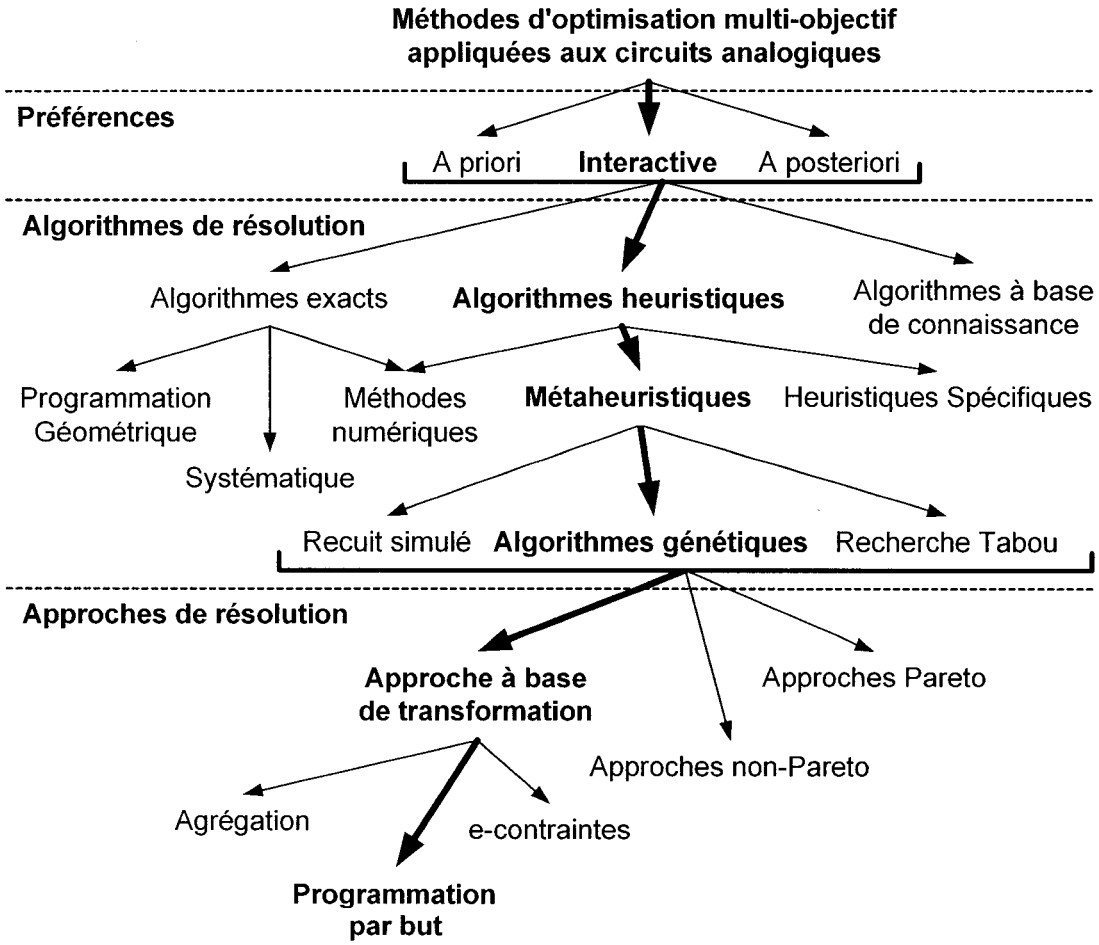


Figure 1.3 Classification des méthodes d'optimisation multi-objectif.

Les mots en caractères **gras** de la Figure 1.3 indiquent les stratégies adoptées à l'intérieur du logiciel d'optimisation présenté dans ce mémoire.

1.1.2 Méthodes à base de transformation du problème vers l'uni-objectif

1.1.2.1 Méthode d'agrégation

C'est l'une des premières méthodes utilisées pour la génération de solutions Pareto optimales. Elle consiste à transformer le PMO en un problème PMO(λ) qui revient à combiner les différentes fonctions de coût f_i du problème en une seule fonction objectif F, généralement de façon linéaire. Certaines solutions Pareto optimales peuvent être obtenues par la résolution du problème mathématique suivant :

$$PMO(\lambda) \Rightarrow \begin{array}{l} \text{minimiser } F = \sum_{i=1}^{n} \lambda_i f_i(x) \\ \text{sachant que } \quad x \in \Re \end{array} \qquad (1)$$

où les poids λ_i sont définis comme étant compris entre $[0,1]$ et dont $\sum_{i=1}^{n} \lambda_i = 1$ avec n égale au nombre de fonctions de coût. Les solutions trouvées à cette optimisation sont dites *supportées* (Figure 1.4).

Lorsque le décideur ne possède pas une bonne connaissance du domaine à explorer, il est possible de construire un ensemble PO par le lancement de plusieurs optimisations avec des attributions aléatoires des valeurs du vecteur de poids λ. Au terme de cette investigation, un ensemble PO* est déduit pour une combinaison de poids favorisant la découverte des solutions à la frontière Pareto optimales du problème. Une approche où le vecteur de poids est adaptatif en cours d'optimisation et où on prend en compte les préférences d'un décideur peut être trouvée dans [107].

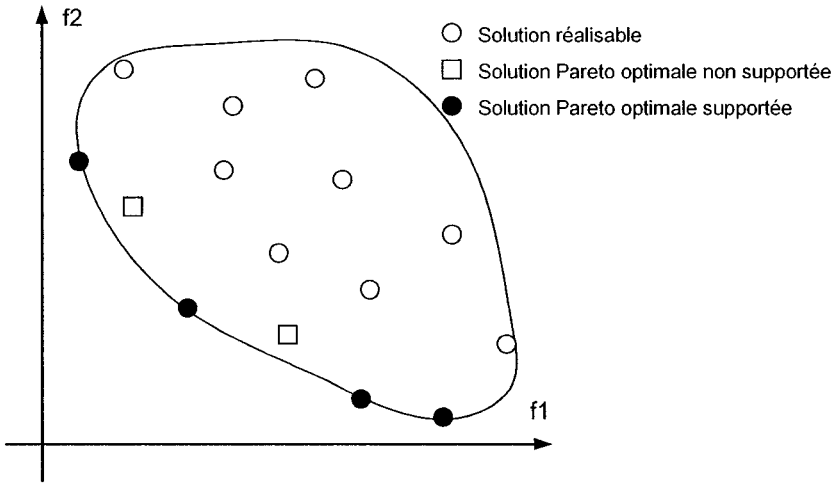


Figure 1.4 Solutions supportées et non-supportées.

La méthode par agrégation n'est pas une méthode intuitive pour le décideur, même si celui-ci connaît les objectifs à atteindre. Par exemple, malgré un ajustement de λ qui semble judicieuse au décideur, il arrive parfois que l'optimisation dérive vers des solutions non-viables à cause des *effets d'ombrage*. Ceux-ci apparaissent entre les fonctions objectifs lorsqu'une mauvaise pondération est imposée. Il en résulte une réduction des contributions normalement apportées au pointage par certaines performances clés. Le circuit converge donc vers un optimum favorisant seulement la performance privilégiée au détriment des autres performances affaiblies. C'est pourquoi, pour le décideur, il devient très difficile de jauger correctement le vecteur de poids qui permet à chacune des performances de s'exprimer sans pour autant enterrer les autres. Même après plusieurs tentatives d'optimisation et de rectification du vecteur de poids, un décideur expérimenté est rarement assuré de posséder la combinaison qui mène l'optimisation vers la *région désirée* de la frontière Pareto optimale.

1.1.2.2 Programmation par but

Dans cette méthode, le décideur doit définir les buts ou références qu'il désire atteindre pour chaque objectif. Ces valeurs sont introduites dans la formulation du problème, le transformant en un problème uni-objectif. Par exemple, la fonction coût peut intégrer une norme pondérée qui minimise les déviations par rapport aux buts. Le problème peut être formulé de la manière suivante :

$$PMO(\lambda,z) \Rightarrow \begin{array}{l} \text{minimiser } F=\left(\sum_{j=1}^{n} \lambda_j \left| f_j(x)-z_j \right|^p \right)^{\frac{1}{p}} \\ \text{sachant que } \quad x \in \Re \end{array} \qquad (2)$$

où $1 \leq p \leq \infty$, et z est le vecteur de référence[6]. La norme utilisée est la métrique de Tchebychev (L_p-métrique). Généralement p est égal à 2 ; dans ce cas, on a une métrique euclidienne. Si $p = \infty$, l'équation revient à une fonction min-max. Une sélection arbitraire du vecteur de référence peut ne pas être désirable, puisqu'un mauvais vecteur de référence peut aboutir à une solution qui n'est pas Pareto optimale.

Les approches basées sur la programmation par but trouvent une solution non-dominée si le but est choisi dans le domaine réalisable. Cependant, le décideur est chargé de choisir le vecteur but et les poids associés à chaque objectif, ce qui est une tâche difficile lorsque la structure de l'espace de recherche n'est pas connue. Si le domaine réalisable n'est pas facile à approcher, la méthode peut être inefficace.

---

[6] Également nommé vecteur but ou vecteur directeur

Dans ce mémoire, une méthode à base de transformation du problème est utilisée. Cette méthode est fortement inspirée du *concept de la programmation par but* (section 2.4.4.1). Mais avant, nous présentons les différents algorithmes de résolution employés dans la littérature, pour l'optimisation des CA transposée en un problème multi-objectif.

1.2 Algorithmes de résolution appliqués à l'optimisation de circuits analogiques

Cette section présente les différentes méthodes étudiées et/ou employées pour effectuer l'optimisation de CA. D'excellentes revues sur le sujet de l'optimisation de CA [28][13][89][5] mettent en perspective les différentes stratégies adoptées à l'intérieur des outils de synthèse au fil des 20 dernières années.

Depuis ses débuts, l'approche à adopter pour résoudre le problème de l'optimisation des CA est disputée entre trois écoles de pensée.

Il y a d'une part l'école de pensée qui prône *l'approche exacte*, et qui se base sur une *représentation analytique* des CA. Puis, d'autre part, il y a celle qui prône *l'approche heuristique* qui explore par échantillonnage le vaste espace des solutions à l'aide d'un *simulateur de circuits*. Enfin, *l'approche à base de connaissance* utilise des réseaux neuronaux et des bases de données pour *aiguiller* l'optimisation vers des cellules analogiques précaractérisées.

L'approche analytique souffre généralement d'un grand coût associé à un *haut degré d'approximation* sur les composantes électroniques. Quant à l'approche

heuristique, elle est généralement pénalisée par un coût élevé en effort de calcul associé à la phase d'évaluation des circuits candidats. Bien que l'approche à base de connaissance peut être incorporée à la méthodologie de synthèse automatisée des CA, ce mémoire est consacré à l'investigation des méthodes *d'optimisation paramétrique* plutôt que sur les méthodes de *reconnaissance de patron.*

Afin de disposer d'une meilleure vue d'ensemble des techniques d'optimisation appliquées aux CA, les sections qui suivent présentent une description de chacune des approches retrouvées dans la littérature.

1.2.1 Classe des algorithmes exacts

La classe des algorithmes exacts est caractérisée par la mise en œuvre d'une approche basée sur une représentation mathématique du problème, afin de pouvoir ensuite procéder à une optimisation du CA.

1.2.1.1 Programmation géométrique

Cet algorithme de résolution est catégorisé dans les approches analytiques, car il nécessite une décomposition du circuit sous forme d'équations.

La programmation géométrique existe depuis la fin des années 60 et elle fut appliquée à plusieurs domaines d'ingénierie [12]. Bien que la programmation géométrique soit connue, elle n'est pas pour autant aussi répandue que son proche parent la *programmation linéaire* [26].

De plus, plusieurs percées importantes dans le domaine des *algorithmes d'optimisation à application générale* pour des problèmes non-linéaires et contraints ont contribué à décroître la popularité de la programmation géométrique au cours des dernières décennies.

La programmation géométrique puise ses fondements dans les propriétés remarquables des problèmes à formulation convexe [11]. Parmi les résultats surprenants qui découlent de l'utilisation de la programmation géométrique, on distingue des solutions trouvées de type *optimum global*[7] et résolues avec une *grande rapidité*[8] [39].

1.2.1.1.1 Processus d'optimisation

Le programme géométrique est un problème d'optimisation de la forme :

$$PMO(x) \Rightarrow \begin{array}{lll} \text{minimiser} & f_0(x) & \\ \text{sachant que} & f_i(x) \le 1 & i = 1,\ldots,m \\ & g_i(x) = 1 & i = 1,\ldots,p \\ & x_i > 0 & i = 1,\ldots,n \end{array} \qquad (3)$$

où $f_0,\ldots,f_m$ sont des fonctions *posynomiales*[9] et $g_1,\ldots,g_p$ sont des fonctions *monomiales*[10]. La fonction $f_0(x)$ est la fonction à minimiser. Une des propriétés

---

[7] Optimisation globale : signifie que l'optimisation converge vers le *seul optimum global* de la fonction à optimiser, en conformité avec les contraintes associées au problème.

[8] L'optimisation d'un problème composé de plus de 100 variables et 1000 contraintes peut être réalisée en moins de 1 à 2 secondes sur une station de travail contemporaine.

[9] Fonction de la forme : $f(x) = \sum_{j=1}^{n} c_j \cdot x_1^{\alpha_{1j}} x_2^{\alpha_{2j}} \cdots x_m^{\alpha_{mj}}$ où $x = [x_1,\ldots,x_m]$, $c_j \ge 0$ et $\alpha_{ij} \in R$.

[10] Fonction de la forme : $g(x) = c \cdot x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_m^{\alpha_m}$ où $x = [x_1,\ldots,x_m]$, $c \ge 0$ et $\alpha_i \in R$.

intéressantes des fonctions posynomiales est quelles sont fermées sur l'addition et la multiplication. Ainsi, la multiplication ou l'addition de fonctions posynomiales produit toujours des fonctions fesant parti de l'ensemble des fonctions posynomiales.

Ensuite, le programme géométrique est transposé en un programme linéaire par la conformation $y = \log(x)$ [11]. Cette manipulation permet aux exposants de descendre au niveau des coefficients. Les algorithmes efficaces de la programmation linéaire/quadratique se charge ensuite d'optimiser le problème.

De récents développements dans les techniques d'optimisation applicables à la *programmation quadratique*, comme ceux qui combinent la méthode du *point intérieur* [46] avec la résolution du problème *primal-dual* [104], assurent une convergence vers des solutions optimales précises et en un temps très court.

1.2.1.1.2 Application aux circuits analogiques

Quelques auteurs ont appliqué la programmation géométrique au problème de dimensionnement des CA [39][63].

Il est important de rappeler que l'optimisation par programmation géométrique est de type analytique. C'est pourquoi, il est nécessaire de déterminer la topologie du circuit à optimiser pour préalablement en extraire toutes les expressions analytiques associées à ses contraintes et ses performances. La programmation géométrique n'accepte d'optimiser que des contraintes et des performances sous une formulation convexe [12]. En ce sens, il faut normalement requérir aux habiletés analytiques d'un expert en

conception de circuit afin de correctement pouvoir transposer le problème. Bien sûr, le concepteur doit aussi toujours se rappeler le contexte qui relie la représentation analytique aux restrictions de la formulation convexe. Heureusement, lors de la décomposition des CA en leurs équations équivalentes, il est généralement facile de retrouver un grand nombre de descriptions analytiques compatibles à cette forme particulière.

Un exemple complet et détaillé de conversion d'un *amplificateur opérationnel à deux étages* en un programme géométrique est présenté dans [39]. Les CA contenant des boucles de rétroaction ne constituent généralement pas un obstacle à la programmation géométrique comme présenté dans [19].

La programmation géométrique supporte l'optimisation d'un CA vis-à-vis les *variations paramétriques* [38]. De plus, une *analyse de sensibilité* est disponible à l'intérieur de la programmation géométrique [63]. Cette analyse de sensibilité permet de connaître les performances les plus sensibles aux variations des paramètres à optimiser. Ainsi, il est plus facile d'identifier les performances qui peuvent être problématiques. De plus, en cas d'échec sur la convergence de l'optimisation, cette analyse de sensibilité produit un *certificat d'infaisabilité* [38] qui démontre explicitement les contraintes ou les spécifications sur les performances à relaxer.

Ces multiples bénéfices confèrent à l'optimisation par la programmation géométrique plusieurs avantages indéniables concernant ses développements futurs [10].

1.2.2 Classe des approches heuristiques

L'algorithme d'optimisation heuristique est défini comme étant une méthode qui explore l'espace des solutions d'abord au hasard puis ensuite en tenant compte des découvertes précédemment réalisées.

Tel que présenté à la Figure 1.5, l'algorithme d'optimisation heuristique se décompose en deux modules; un *solveur* dont le rôle est de proposer des circuits candidats et un *simulateur* qui simule le circuit candidat et le convertit en ses performances équivalentes. Une fois ces performances retournées au solveur, une valeur de pointage est assignée au candidat proposé. Ce pointage est déduit des performances atteintes comparées aux spécifications dictées par le concepteur. Par définition, le solveur cherche par *essais et erreurs* les circuits candidats qu'il soupçonne les plus aptes à minimiser la fonction de coût. Lorsque ce processus est mis en boucle, une voie vers la minimisation se trace automatiquement.

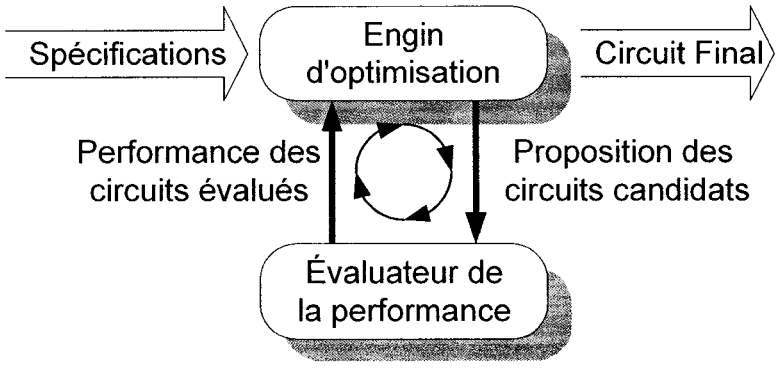


Figure 1.5 Modèle abstrait des outils de synthèse analogique utilisant un simulateur pour sonder l'espace des circuits

La méthodologie de synthèse basée sur l'intervention d'un simulateur de circuits pose de grands défis. Sachant bien que les simulateurs de circuits commerciaux ne sont pas spécialement conçus pour être exécutés à des miliers, voire des dizaines de milliers de reprises, à l'intérieur d'une boucle d'optimisation, il devient nécessaire de recourir à un algorithme de recherche qui soit suffisamment efficace et performant pour tirer profit de l'utilisation d'une simulation complète de chaque circuit candidat proposé par l'algorithme de recherche.

Les sections qui suivent présentent plusieurs algorithmes de recherche heuristiques faisant intervenir, de manière plus ou moins intensive, un logiciel de simulation pour l'évaluation des performances de chaque circuit électronique candidat.

1.2.2.1 Méthode numérique

Les méthodes numériques d'optimisation non-linéaire ont fait l'objet de nombreuses études depuis plus de quarante ans, et il existe beaucoup de littérature sur ce sujet allant de l'analyse numérique à la recherche opérationnelle. Bien sûr, cette discipline adresse normalement des problèmes de nature mathématique, mais de récents progrès dans ce domaine ont donné naîssance à des méthodes plus évoluées telles que la méthode par gradients conjugués [66] et les algorithmes de Levenberg-Marquardt [61].

On associe généralement les méthodes numériques aux algorithmes d'optimisation par la méthode de descente de gradient [11], les méthodes de Newton [85][27], la méthode des gradients conjugués [67][66] et la méthode du Simplex [74][43].

Un des grands handicaps associés de ces méthodes est *leur totale dépendance à l'évaluation des dérivées partielles* sur les fonctions objectifs. Ces dérivées partielles sont exigées puisque, sans elles, ces méthodes ne peuvent converger.

Pour les fonctions d'objectifs avec une dérivée seconde continue, trois familles générales d'algorithmes ont été trouvées comme étant efficaces sur une large gamme de problèmes pratiques :

- Pour un **faible** nombre de variables, l'algorithme de *Newton stabilisé* et l'algorithme de *Gauss-Newton* sont efficaces. Ils incluent aussi normalement divers algorithmes comme celui de *Levenberg-Marquardt* [65] ou celui d'établissement de *régions de confiance* [23][16].
- Pour un nombre **modéré** de variables, les algorithmes *Quasi-Newton* sont appropriés et efficaces.
- Pour un nombre **élevé** de variables, il est préférable de faire appel aux algorithmes à *gradients conjugués*.

Le calcul du Jacobien ou du Hessien est généralement chose commune pour ces méthodes. Cependant, il est aussi connu que le calcul des dérivées premières et secondes d'un système complexe multi-variable comme un CA peut être une tâche très ardue. De plus, ces algorithmes d'optimisation ne sont pas adaptés pour naviguer dans des espaces de recherche bruités, discontinus ou non-dérivables.

Toutes les méthodes mentionnées ci-dessus convergent habituellement vers des *optima locaux*. Elles ne garantissent jamais de trouver un optimum global. En pratique,

l'algorithme de Levenberg-Marquardt se distingue en trouvant généralement de meilleurs optima que les autres méthodes usuelles pour une large variété de problèmes.

Pour réaliser une optimisation globale, à l'aide d'une des méthodes énumérées précédemment, il faut normalement procéder à un grand nombre de lancements de l'optimisation avec des points de départ différents et choisis de manière aléatoire. C'est pourquoi, lorsque c'est possible, on privilégie l'intégration d'un point de départ correspondant à une solution de départ acceptable. De cette manière, la convergence vers un minimum environnant acceptable est grandement favorisée.

1.2.2.1.1 Application aux circuits analogiques

On retrouve occasionnellement, intégré à l'intérieur des simulateurs de circuit, un module d'optimisation multi-variable basé sur une des méthodes numériques présentées. L'algorithme le plus couramment utilisé est l'algorithme de Levenberg-Marquardt (LM). On le retrouve intégré dans certains grands simulateurs de circuit dont HSpice [108]. Par contre, on retrouve plutôt dans le simulateur Pspice une version Newton modifiée comme algorithme d'optimisation. Les simulateurs de circuits peuvent réaliser, moyennant un effort supplémentaire, le calcul des dérivées premières et secondes d'un circuit autour de son point d'opération par des méthodes comme celles des *différences finies* ou de *sensibilité* [77]. La nature continue et dérivable des composantes électroniques analogiques est mise à profit dans la réalisation de cette tâche.

Parmi les problèmes recensés sur les algorithmes numériques, on découvre une grande sensibilité face au *point de départ* sélectionné et une certaine difficulté à gérer une *grande dimensionnalité* avec un nombre croissant de variables et d'objectifs.

Pour les applications en électronique analogique, les méthodes numériques sont très répandues, mais elles sont aussi souvent rapidement limitées en matière de complexité. Les optimisateurs intégrés ne constituent pas un choix de prédilection pour l'optimisation des CA, à cause de leur trop forte propension à facilement se piéger dans les minima locaux. Bien que la nature des CA soit continue, les calculs associés aux dérivées premières et secondes du circuit nécessitent quand même un effort de calcul appréciable, sans offrir pour autant de garanties supplémentaires sur la convergence globale.

1.2.2.2 Algorithme génétique

L'algorithme génétique (AG) est une approche évolutive qui tente d'imiter le processus d'optimisation qui permet aux organismes vivants de s'adapter, de survivre et de performer dans leurs environnements. Les premiers travaux sur les AG ont été réalisés par Holland [40] en 1975.

Afin de bien imiter le processus d'évolution, les AG utilisent une représentation du problème à optimiser sous la forme de gènes.

Dans le cas de l'optimisation de CA, le gène consiste en une chaîne de valeurs binaires nommé le *génotype* (Figure 1.6). Le génotype, dans la nature, représente les

séquences d'acides aminés présentes dans nos gènes. Ensuite intervient le processus de *traduction*, qui convertit le gène binaire en des valeurs numériques (*phénotype*). Le phénotype représentent, dans le cas des CA, les dimensions des composants électroniques à optimiser. Bien sûr, pour conserver la précieuse correspondance entre le génotype et le phénotype, l'étape de traduction ne doit pas être altérée au cours du processus d'optimisation.

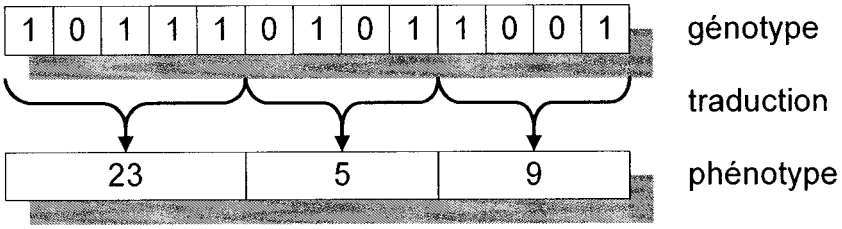


Figure 1.6 Représentation et traduction du gène en variables.

L'AG peut effectuer certaines opérations sur les gènes à l'aide de trois opérateurs de base : la sélection naturelle, la reproduction et la mutation.

La *sélection naturelle* offre la possibilité, au candidat le plus performant, de se reproduire. Cette opération permet ainsi à un individu, plus performant que la moyenne, de multiplier la probabilité de pouvoir dissimer son patrimoine génétique à travers la population. La reproduction par la *recombinaison du bagage génétique* (Figure 1.7) permet à deux parents jugés performants de perpétuer, à travers leurs progénitures, certains *traits génétiques* ayant favorisés leur survie. La *mutation* permet à la recherche de préserver une certaine *diversité génétique*, tout en permettant à une population d'individus de pouvoir échapper aux pièges des minima locaux. Enfin, tout comme

l'évolution, l'AG ne mise pas *uniquement* sur une seule solution, mais il mise plutôt sur une *population de solutions*.

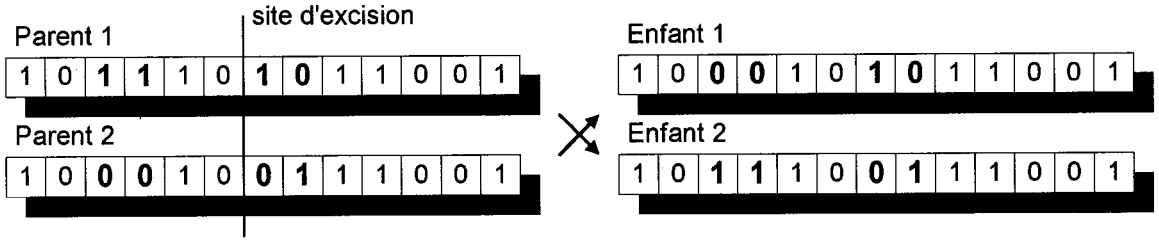


Figure 1.7 Représentation simplifiée de la recombinaison génétique.

1.2.2.2.1 Processus d'optimisation

La Figure 1.8 présente de manière simplifiée la séquence suivie par le processus d'optimisation basé sur les algorithmes génétiques ou évolutifs.

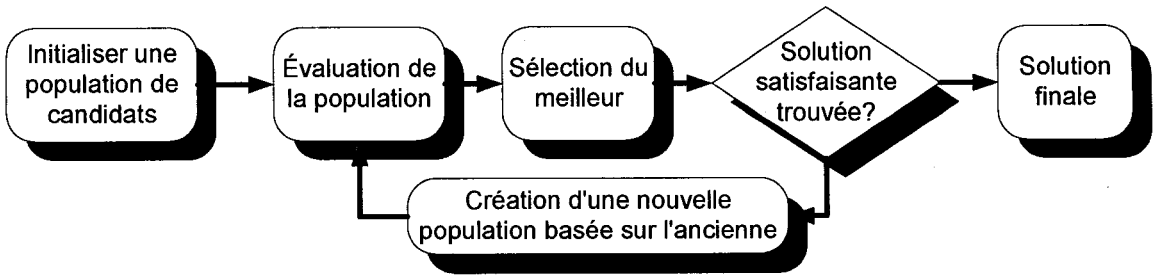


Figure 1.8 Processus d'optimisation simplifié pour les algorithmes génétiques.

La Figure 1.9 présente avec plus de détails les opérateurs génétiques utilisés lors de la création de la nouvelle population entre chaque génération.

Les opérateurs, de même que toutes les décisions réalisées pour chacun de ces opérateurs, s'appliquent toujours par rapport à une distribution de probabilité uniforme. En fait, face à une décision où toutes les possibilités sont équivalentes, l'AG laisse toujours le hasard trancher sur le choix.

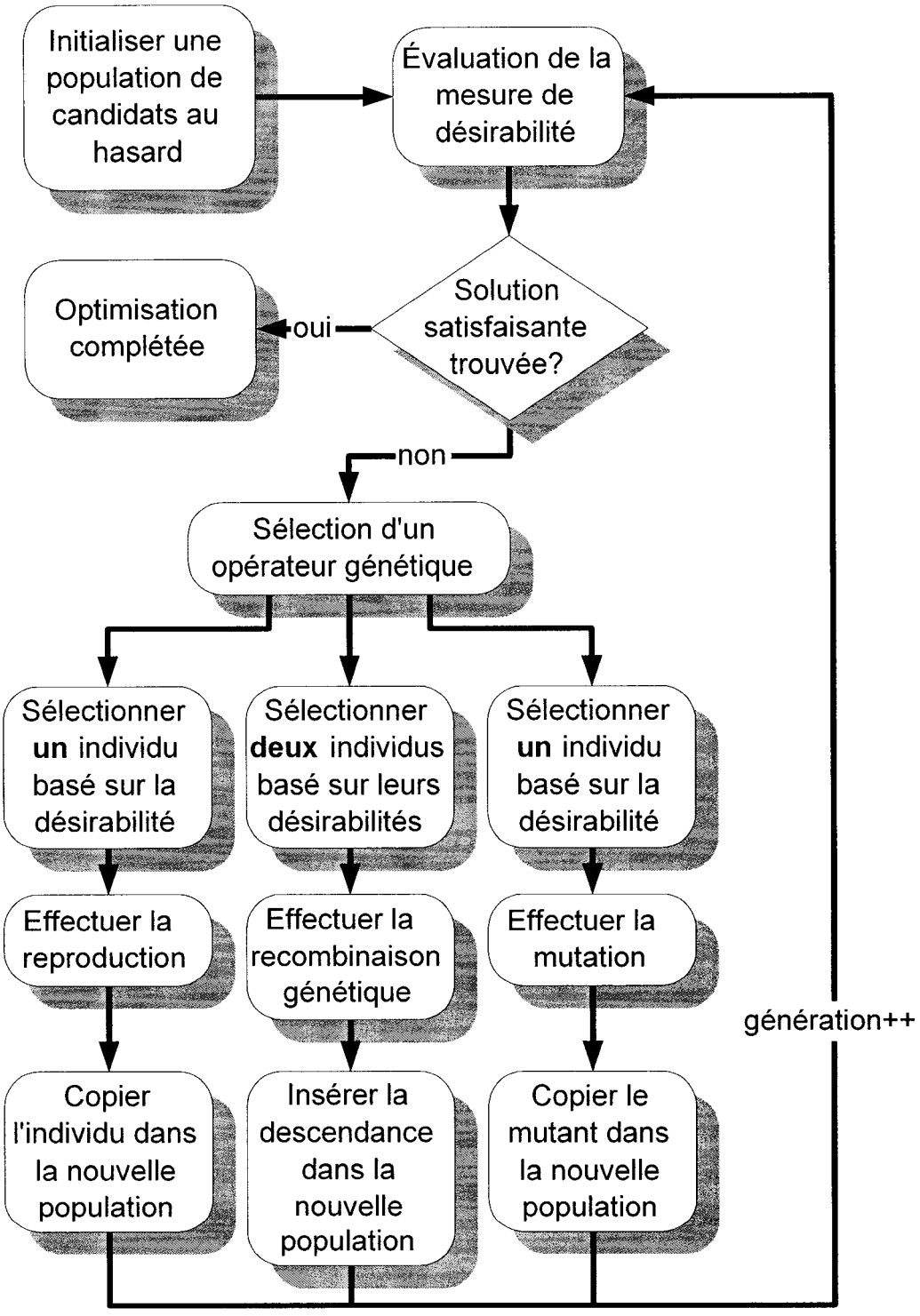


Figure 1.9 Détails du processus d'optimisation de l'algorithme génétique.

1.2.2.2.2 Application aux circuits analogiques

Plusieurs publications traitent de l'utilisation des AG pour réaliser la tâche d'optimiser des CA. En particulier, dans les articles suivants [102][22][106][107], l'optimisation d'amplificateurs opérationnels par l'approche génétique est investiguée.

Comme la plupart des algorithmes métaheuristiques, les AG ne nécessitent aucune connaissance *a priori* du domaine de l'électronique analogique, puisque la conversion des circuits candidats en leur valeur de désirabilité[11] est prise en charge par le simulateur de circuit. Ainsi, étant donné la versatilité des simulateurs de circuits contemporains, rien ne limite l'optimisateur à une catégorie stricte ou particulière de circuits. De cette manière, d'autres types de circuits comme les références de tension [91][72] ou des amplificateurs bas-bruit [99] ont été optimisés par cette même approche.

1.2.2.3 Programmation génétique

La programmation génétique est en plusieurs points semblables aux AG, dans le sens où elle utilise les mêmes concepts d'évolution. Cependant, la programmation génétique diffère sur la *représentation des gènes* et leur interprétation. Contrairement aux AG où le gène encode *une solution paramétrique* du problème à optimiser, la programmation génétique encode dans le gène *le programme devant résoudre* le problème à optimiser. La représentation des données la plus appropriée pour accomplir cette tâche est une *architecture arborescente*. L'arbre génétique peut être décodé selon la

[11] Pointage ou coût

méthode en *profondeur* (*top-down, bottom-up*) ou en *largeur* (*breath-first, depth-first*) [90].

1.2.2.3.1 Processus d'optimisation

Le premier à investiguer l'application de la programmation génétique à l'évolution de CA est Koza [51] en 1997. La majeure partie du processus d'optimisation de la programmation génétique (PG) demeure semblable aux AG (section 1.2.2.2) [47]. La Figure 1.10 présente le corps de la fonction d'évaluation pour une optimisation appliquée au domaine des CA.

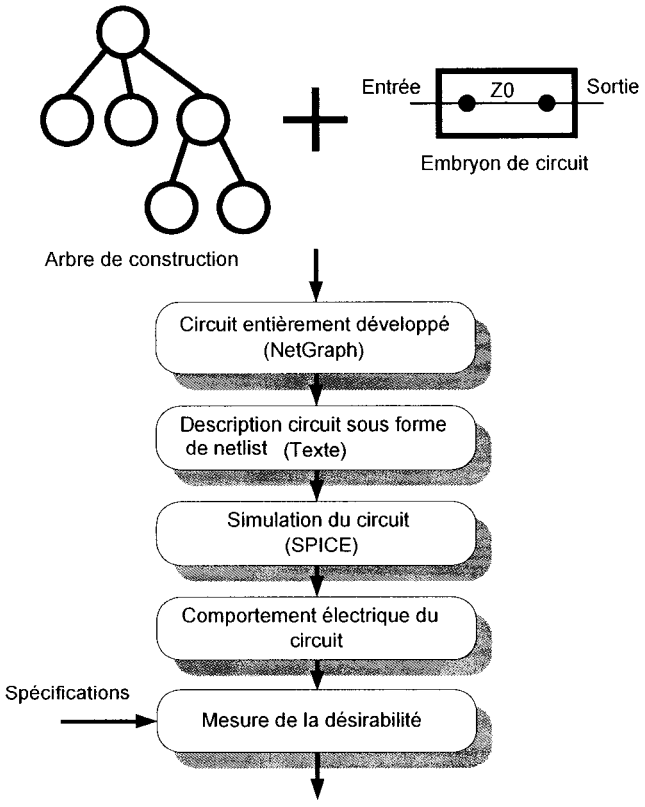


Figure 1.10 Étapes de traitement suivies par la fonction de coût pour évaluer la désirabilité. d'un CA candidat

On remarque que la procédure suivie pour traduire un candidat de solutions, de sa forme arborescente jusqu'à une valeur de désirabilité, passe par l'évaluation via un simulateur de circuit. Pour accomplir cette tâche, certaines transformations sur les données sont nécessaires. Il faut d'abord appliquer l'*arbre de construction* sur l'*embryon de circuit*, afin de produire le graphe du circuit équivalent. La Figure 1.11 offre une représentation visuelle de cette conversion. Ensuite, le *graphe d'interconnexion* correspondant au circuit (NetGraph) est converti en une description textuelle nommée une *netlist*. La *netlist* est le seul format de description de circuit que les simulateurs peuvent accepter. Enfin, on termine l'évaluation par la capture des résultats de la simulation pour établir la valeur de désirabilité atteinte par ce candidat.

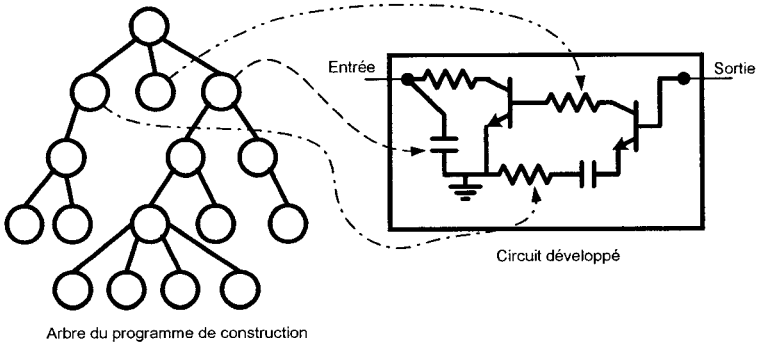


Figure 1.11 Illustration de la conversion de l'arbre génétique de construction en un circuit.

Chacun des nœuds rencontrés lors de la procédure de traduction effectue une modification sur le programme solveur, qui est en l'occurrence ici le CA. Les nœuds peuvent contenir différentes opérations de modification sur le circuit *en cours de développement*. Par exemple, il peut s'agir d'un nœud qui *effectue une opération topologique* (ajout, retrait, sérialisation, parallélisation ou permutation d'une

composante). Il peut aussi s'agir d'un nœud qui *fixe la valeur paramétrique* d'une composante. Et enfin, il peut s'agir d'un nœud qui contient une *instruction de contrôle* (fin de développement du circuit (instruction END) ou absence d'opération (instruction NOP)).

Les règles de syntaxe et de grammaire liés à la procédure de traduction doivent toujours être bien définies et suivies à la lettre afin de préserver l'intégrité structurelle des circuits traduits [4].

En résumé, la programmation génétique permet d'évoluer l'*architecture de la solution* à un problème en même temps qu'elle résout le dit problème. C'est ce qui rend la programmation génétique si exceptionnelle en soi.

1.2.2.3.2 Application aux circuits analogiques

Dans [50], on présente une multitude de circuits passifs inventés par la programmation génétique. On y présente, entre autres, certaines redécouvertes reliées à des brevets existants sur les filtres : Butterworth, Chebychev et Elliptiques. Il est parfois même possible qu'une optimisation réalisée par la programmation génétique conduise à une amélioration des brevets existants [53].

En conclusion, la programmation génétique est une des rares approches qui réalise une véritable exploration de l'espace des topologies de circuits; d'une manière totalement libre, souvent non-conventionnelle et parfois contre-intuitive pour l'humain. Deux très bons exemples pour illustrer le potentiel novateur derrière cette approche sont relatées

dans [53][9]. Cette stratégie peut ainsi conduire à l'émergence de topologies de circuits encore jamais investiguées auparavant et pourtant bien fonctionnelles.

Il est généralement plus courant de voir la programmation génétique reconverger rapidement vers des topologies apparentées aux découvertes de l'Homme [52][48]. Mais lorsque c'est possible, la programmation génétique réussit à *enrichir* la synthèse d'un circuit par l'intervention de judicieuses altérations (vis-à-vis la topologie traditionnelle reconnue) améliorant, par le fait même, les performances désirées.

1.2.2.4 Recuit Simulé

Le Recuit Simulé (RS) est une approche *stochastique* qui appuie ses fondements sur un phénomène physique [88][97][44]. Le principe découle de l'analyse du comportement d'un système physique qui tend à se *cristalliser* vers un niveau enthalpique minimisé lorsque la température diminue. Concrètement, le meilleur parallèle disponible est affilié au domaine de la physique des matériaux. Un alliage métallique composé de deux types de métaux en concentrations données va tendre à minimiser les stress internes liés à la disposition des atomes dans le réseau cristallin. Pour y parvenir, on soumet l'alliage métallique à plusieurs phases de chauffage rapides suivies de phases de refroidissement progressif. Sous l'effet de la chaleur, les phases de chauffage permettent aux atomes de l'alliage de se mouvoir grâce à l'agitation thermique accrue. Puis à mesure que la température diminue, les atomes disposent de moins en moins d'énergie pour se déplacer d'un site à un autre à l'intérieur du réseau cristallin. Ainsi, après plusieurs

recuits, l'organisation des atomes tendra vers un point d'équilibre présentant un minimum énergétique.

1.2.2.4.1 Processus d'optimisation

La seule variable de contrôle du recuit thermique est une pseudo-température. Cette température est un indicateur direct de l'amplitude des perturbations pouvant affecter les paramètres optimisables. Comme tous les autres algorithmes de recherche de la classe des heuristiques, le recuit simulé dispose d'un mécanisme qui lui permet de s'échapper de l'emprise des minima locaux.

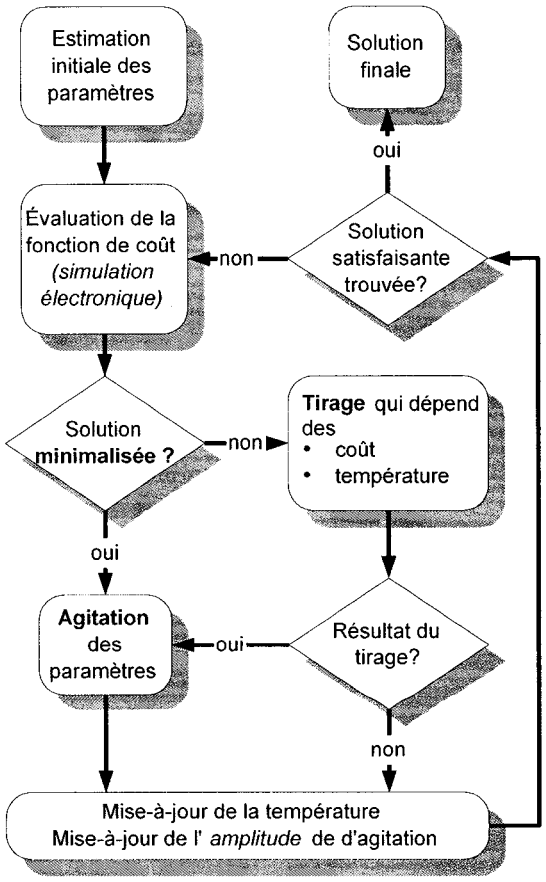


Figure 1.12 Processus d'optimisation du recuit simulé.

Le processus simplifié de cette approche est présenté à la Figure 1.12. Les paramètres optimisables sont initialement sélectionnés de manière aléatoire. À chaque itération, un circuit est proposé par l'algorithme de recuit et ce circuit est évalué pour déterminer la valeur du coût associé aux performances atteintes. Normalement, moins un circuit remplit les exigences des spécifications et plus son coût sera élevé. Il est bien entendu que l'optimisation cherchera à minimiser ce coût.

Suite à l'évaluation de la solution courante, une amélioration du coût de celle-ci s'enchaîne par une perturbation des paramètres, afin d'explorer une nouvelle solution environnante. Lorsque la fonction de coût ne se trouve pas améliorée, deux choix s'offrent à la recherche : le *status quo* ou persister dans la perturbation des paramètres optimisables. Ce choix est déterminé de manière aléatoire et est basé sur une distribution de probabilité modulée par deux facteurs importants. Ces deux facteurs sont la valeur de la *température courante* et la valeur de *coût associée à la solution courante*. À température élevée, la perturbation des paramètres est favorisée, tandis que lorsque le coût associé à la solution courante est faible, le cas contraire se produit. Enfin, malgré un tirage optant pour le *status quo*, l'optimisation poursuit son cours normal en effectuant la mise à jour du paramètre de contrôle de la température selon le principe exponentiel dégressif présenté à la Figure 1.13.

Afin de bien reproduire le processus naturel de recuit, la pseudo-température agissant sur les paramètres est progressivement réduite au fur et à mesure que les itérations se succèdent. Un parallèle peut être établi entre l'amplitude des perturbations

sur les paramètres optimisables et l'agitation thermique des atomes d'un matériau. Ainsi, plus la température diminue et plus la solution tend à se cristalliser.

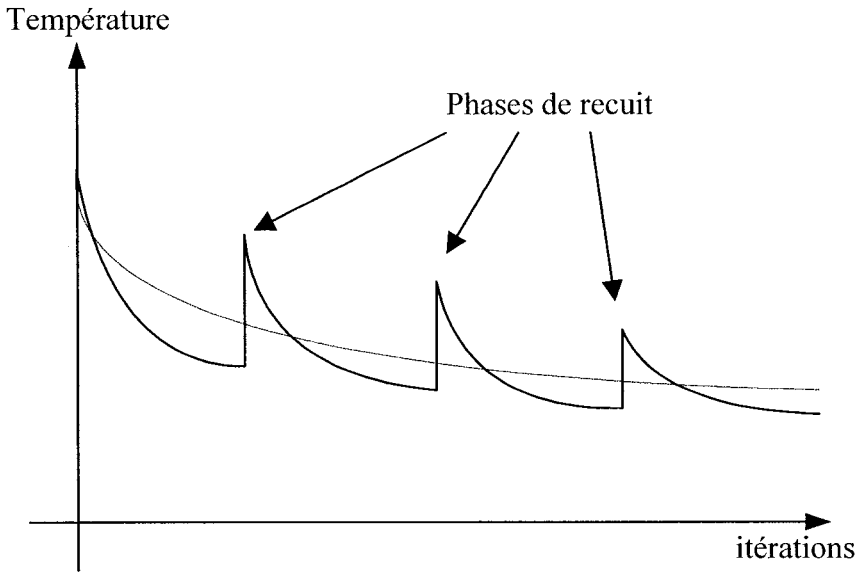


Figure 1.13 Exemple de contrôle de la température pour les fins de recuit simulé.

Le recuit simulé fonde sa recherche sur un unique candidat échantillonneur/explorateur de l'espace des solutions. Étant donné qu'on utilise un seul candidat pour réaliser l'exploration de l'espace des solutions, il n'est pas rare que ce dernier se retrouve piégé dans un minimum local lors des premiers cycles thermiques. C'est pour cette raison que des cycles rapides de passage par les hautes températures sont incorporés dans le contrôle de la température. Ces chauffages abrupts fournissent, ponctuellement, suffisamment d'énergie à la solution courante pour se dégager du minimum local.

Enfin, le recuit simulé est une méthode généralement reconnue pour sa robustesse face aux fonctions de coût bruitées ou même discontinues.

1.2.2.4.2 Application aux circuits analogiques

On retrouve dans la littérature plusieurs applications de cette approche [103][68][96] et elle a été utilisée dans plusieurs outils de synthèse analogique pour l'optimisation paramétrique des circuits [43][79][95].

Les reproches généralement faits à cette méthode sont sa très grande gourmandise en temps de calcul lié au *nombre d'appel à la fonction de coût* et sa *convergence non-déterministe*. Puisque les évaluations sont réalisées par l'intermédiaire d'un simulateur de circuit, la convergence peut rapidement nécessiter des *efforts de calcul* pouvant s'échelonner sur plus d'une semaine [79]. Le recuit simulé est largement utilisé lorsque qu'aucun autre algorithme n'est disponible et que l'on désire procéder à une *optimisation globale* sur la fonction de coût malgré le grand effort de calcul qu'on lui connaît.

Parmi les variantes de cette approche, il est possible d'adapter l'algorithme du recuit simulé de sa forme utilisant un seul candidat vers une approche multi-candidat [105]. Cette stratégie permet surtout de multiplier la probabilité de trouver un optimum mais entraîne aussi une multiplication du temps de convergence à cause du nombre supplémentaire d'évaluations encourues.

1.2.2.5 Recherche Tabou

La Recherche Tabou (RT) a été initialement proposée par Glover en 1977 comme une approche heuristique pouvant résoudre des problèmes d'optimisation non-linéaires [31]. Un supplément de détails sur les principes de fonctionnement de la RT peuvent être trouvés dans [35].

La caractéristique principale de l'heuristique RT est son habileté à pouvoir s'échapper des minimums locaux par la maintenance d'une liste de mouvements dits *interdits*. Cette approche converge vers d'excellents résultats tout en faisant appel à un faible nombre d'itérations, privilégiant ainsi l'utilisation d'un simulateur de circuit à l'étape de l'évaluation de la fonction de coût.

Les sections qui suivent présentent brièvement les fondements de la Recherche Tabou.

1.2.2.5.1 Processus d'optimisation

Afin de bien cerner la nature et le principe de fonctionnement de l'algorithme par RT, il est nécessaire de recourir à une représentation graphique simplifiée de son processus.

Comme un grand nombre de méthodes d'optimisation qui utilisent une séquence de déplacements dans l'espace des solutions pour converger vers un minimum, la RT sonde l'espace de recherche de manière itérative, par un point échantillonneur P qui se promène d'un essai de solution à l'autre. La destination du prochain point exploratoire P_{t+1} est toujours choisit en tenant compte de l'historique des t derniers points explorés. En résumé, le prochain point est restreint au *voisinage permis* autour de la solution courant P et ne doit *jamais* entrer en conflit avec une des anciennes entrées de la liste tabou.

La classe des heuristiques par *descente de gradient* n'accepte généralement que des déplacements qui conduisent à une réduction de la fonction de coût. Ces méthodes sont capables de trouver un *minimum*, mais elles n'offrent aucune garantie que celui-ci soit un *minimum global* sur la fonction de coût.

La Recherche Tabou intègre des éléments additionnels à l'intérieur de l'heuristique, afin de guider et d'encourager la recherche à poursuivre son exploration de l'espace des solutions, malgré une suite cumulée de déplacements n'améliorant pas la fonction de coût. Sans la possibilité de rebrousser chemin, il devient ainsi impossible de retomber dans le minimum local duquel l'algorithme vient justement de s'échapper.

À cette fin, un *ensemble tabou* T est créé à l'intérieur duquel, les éléments présents constituent des balises aux *déplacements interdits*. Les éléments de la liste tabou résument l'historique du processus de recherche jusqu'à t itérations dans le passé. L'objectif de la liste tabou est de marquer les régions de l'espace qui ont déjà été visitées.

L'exemple de la Figure 1.14 illustre, dans un espace unidimentionnel, la stratégie d'échappement à un minimum local. Lorsque l'entrée $T[3]$ est ajoutée à la liste tabou, il devient impossible au prochain point de se retrouver sous la région couverte par $T[1]$, $T[2]$ et $T[3]$, c'est ce qui l'oblige à continuer en direction de $T[4]$. À l'étape de $T[6]$, le point P_2 ne peut être retenu à cause de son status de conflit avec la liste tabou vis-à-vis les entrées $T[5]$ et $T[6]$.

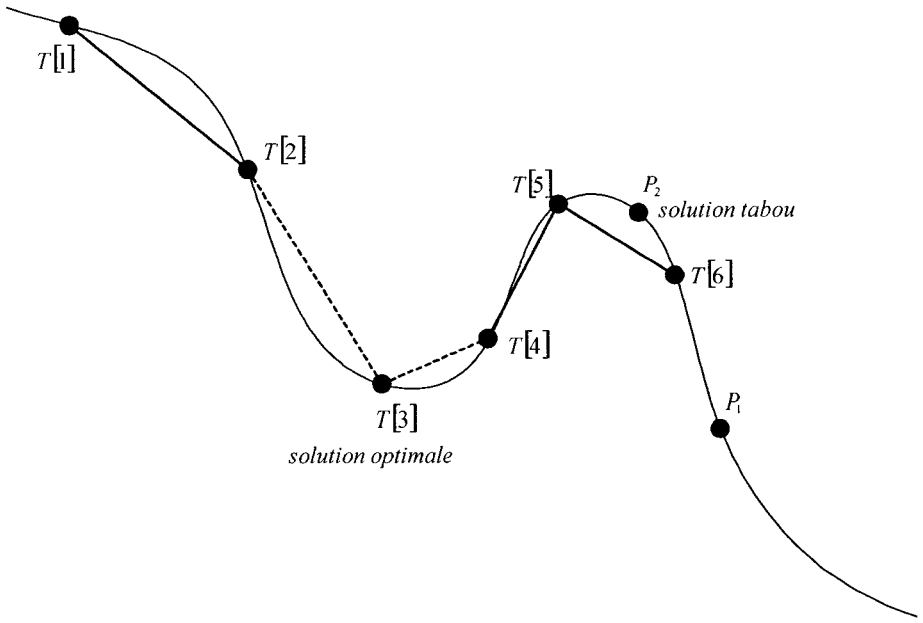


Figure 1.14 Mécanisme d'échappement des minimums locaux (cas unidimentionnel).

Le statut de légalité d'une solution exploratoire est vérifiée par l'inégalité suivante:

$$dist(P,T[i])^2 + dist(P,T[i-1])^2 < dist(T[i],T[i-1])^2 \qquad \exists i, \quad 2 < i < t, \qquad (4)$$

Si cette inégalité est respectée, alors le point exploratoire proposé se retrouve dans la région interdite. Il doit donc, par conséquent, être écarté des déplacements futurs possibles. Une représentation visuelle du statut de légalité entre la liste tabou T et deux points P et P' candidats sont présentés à la Figure 1.15:

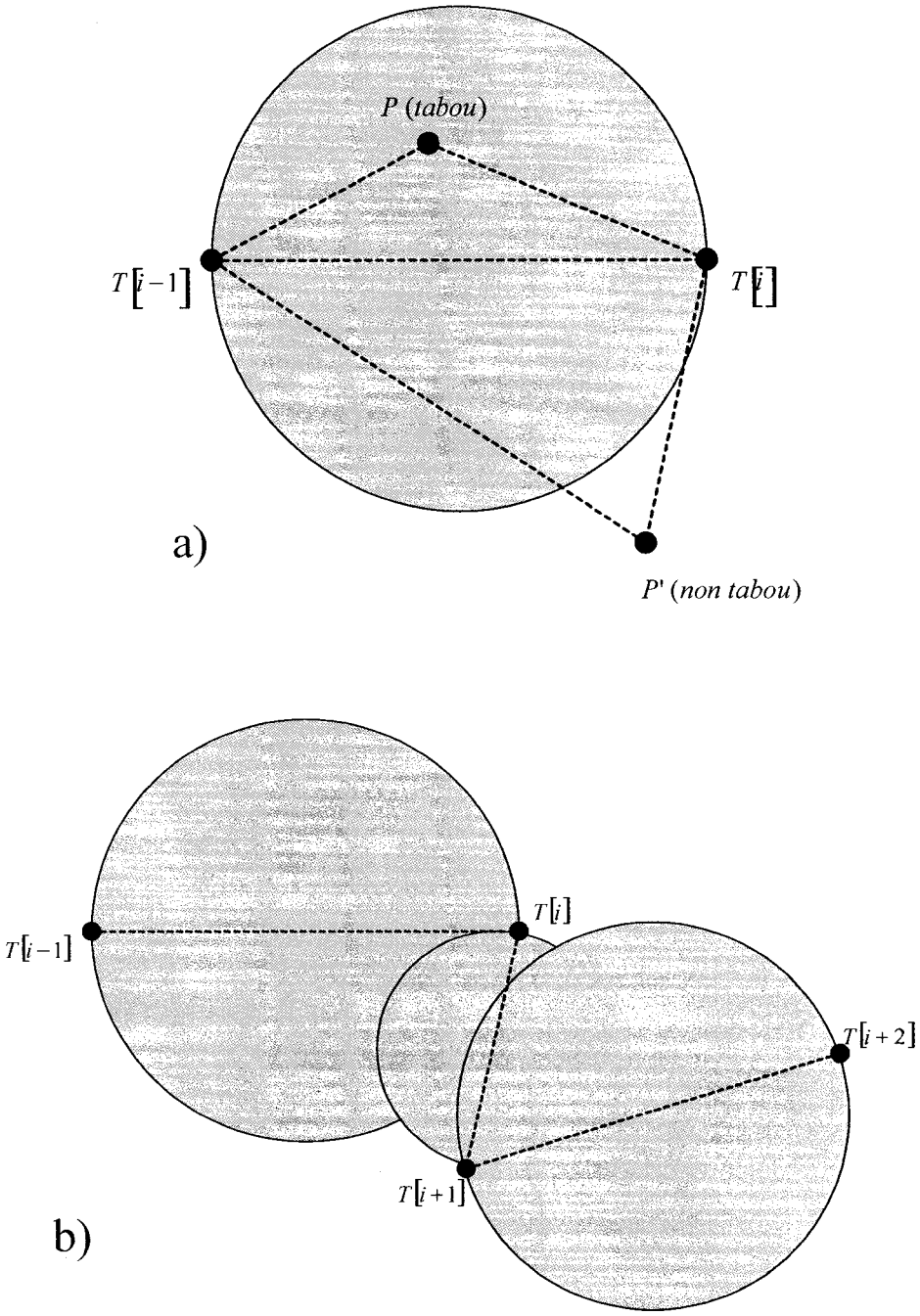


Figure 1.15 Status de légalité de futurs points en regard de la liste tabou (a) et aire couverte par la liste tabou (b) (cas bidimensionnel).

La procédure de Recherche Tabou est décrite par les étapes présentées à la Figure 1.16.

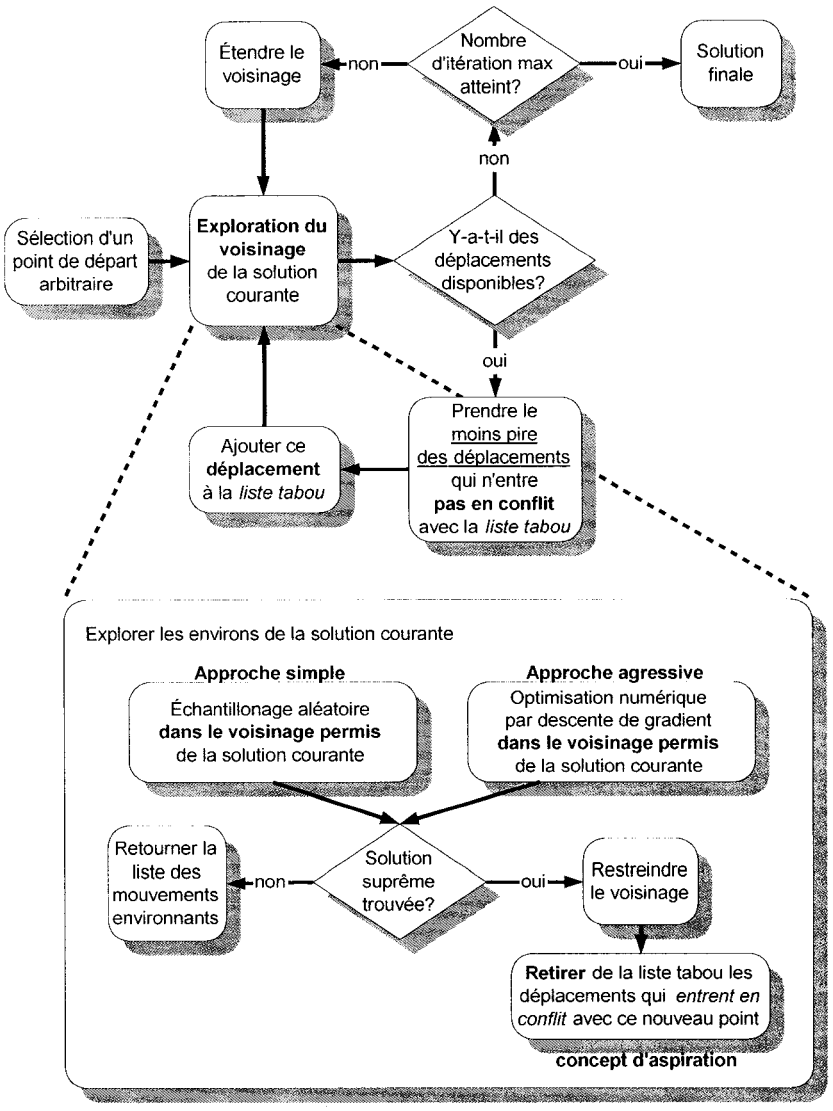


Figure 1.16 Processus d'optimisation employé par la Recherche Tabou.

Un choix simple pour la fonction d'*exploration du voisinage* consiste à utiliser une stratégie d'échantillonnage en générant aléatoirement n déplacements dans *le voisinage de la solution courante* et en prenant celui qui possède le coût minimal.

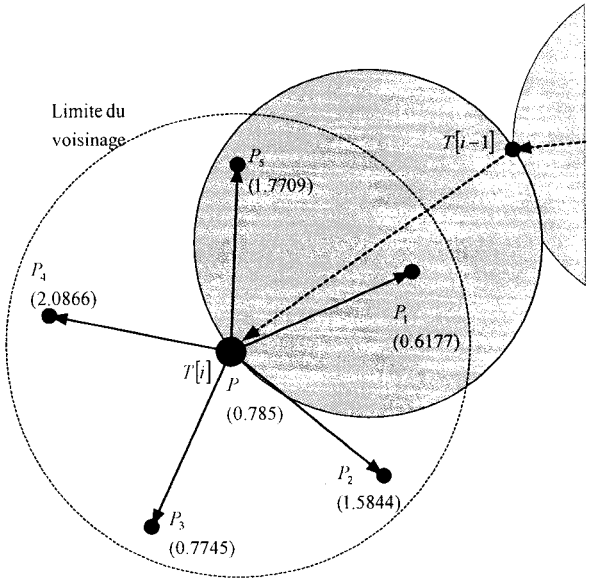


Figure 1.17 Mécanisme de sélection de la méthode Tabou.

À chaque exécution de cette étape, on sélectionne le déplacement qui offre la meilleure amélioration de la fonction objectif f, assujettie à la restriction que seuls les mouvements non-tabous sont permis. Si tous les déplacements disponibles mènent à de mauvaises solutions, alors on sélectionne la meilleure de cet ensemble.

Dans ce cas, la RT consiste en une séquence de minimisations locales qui privilégie le *meilleur mouvement disponible* à chaque itération. Cette approche contraste avec les méthodes qui convergent lentement vers les minima de la fonction d'objectif tel que la technique du recuit simulé.

Le but principal de la liste tabou T est d'éviter le repassage par une solution déjà visitée. En écartant de la sélection les *déplacements* contrevenants à un des mouvements effectués dans une séquence de t itérations antécédentes, la procédure de déplacements s'éloigne progressivement de toutes les solutions parcourues au cours des t dernières

itérations. Pour t suffisamment grand, la vraisemblance de repassage par des solutions visitées s'évanouie presque totalement.

Le dernier concept clé, propre à la RT, est celui de la *phase d'aspiration* (Figure 1.16). Lorsqu'une exploration du voisinage est réalisée par la *fonction d'exploration*, il est possible de découvrir une ou plusieurs solutions qui sont *meilleures que toutes les solutions rencontrées* depuis le démarrage de l'optimisation. Dans ces circonstances, le concept d'aspiration prend effet en emmagasinant cette nouvelle solution comme étant la meilleure solution, puis l'algorithme retire de sa liste tabou l'entrée qui interdisait ce déplacement. Le concept d'aspiration permet ainsi à une optimisation de pouvoir revenir sur ses pas l'instant d'une exploration locale *si et seulement si une solution meilleure à toutes celles rencontrées est découverte*.

1.2.2.5.2 Application aux circuits analogiques

Une seule application de cet algorithme d'optimisation a été trouvée dans la littérature [1]. Cette méthode se distingue des autres par son côté très économe dans le nombre d'appels nécessaires au simulateur de circuit afin d'atteindre la convergence. Dans un exemple de circuit contenant 4 paramètres libres, il est possible de réduire, par un facteur de 10 à 100 fois, le nombre d'appels à la fonction de coût comparativement à la méthode par recuit simulé [7]. Cependant, la RT ne préserve pas cette impressionnante convergence lorsque le nombre de variables libres croît. C'est probablement pour cette raison que les circuits synthétisés par l'auteur ne dépassaient guère plus de 5 variables indépendantes.

1.2.3 Autres approches

1.2.3.1 Approches hybrides

Sans nous attarder sur les méthodes hybrides, celles-ci concernent essentiellement les méthodes d'optimisation qui combinent plus d'une méthode, comme celles présentées dans les sections précédentes. Par exemple, une méthode par AG, qui sonde de manière globale l'espace des solutions, en combinaison avec une méthode par descente de gradient, qui effectue la découverte des optima locaux, se révèle être une méthode hybride qui tire le meilleur profit des atouts de chacun des deux algorithmes d'optimisation. Il va sans dire que plusieurs autres types d'algorithmes peuvent être combinés ensemble pour produire une méthode d'optimisation hybride. Plusieurs outils de synthèse analogique ont opté pour cette voie dans la résolution optimale des paramètres des circuits (section 1.3).

La recherche Tabou appliquée à l'optimisation de CA [1] peut utiliser la propriété continue de l'espace des solutions de CA en utilisant une forme de descente de gradient [8] dans sa phase d'investigation du voisinage de la solution courante (Figure 1.16).

Il est aussi possible d'adapter le recuit simulé avec des AG pour en arriver à une approche hybride présentée dans [54].

L'outil de synthèse analogique FASY [95] utilise un algorithme d'optimisation qui combine en deux phases le recuit simulé et la méthode par gradient conjugué Fletcher-Powell.

L'outil OPTOMEGA [43] adopte aussi une stratégie croisée en utilisant le recuit simulé et la méthode Simplex, selon les préférences du coordonnateur de l'optimisation.

1.3 Éventail des outils académiques et industriels

L'intérêt de ce mémoire est principalement orienté sur le problème de l'optimisation des CA, pour lequel plusieurs travaux ont été réalisés depuis quelques décennies. Dans cette section, on présente quelques logiciels d'optimisation disponibles sur le marché, aussi bien de provenance académique, qu'industrielle.

En ce qui concerne le Tableau 1.1, les différents outils relatés dans la littérature scientifique sont présentés. Ce tableau présente la stratégie sur laquelle s'appuie l'optimisation ainsi que la philosophie à laquelle elle se rapporte : à base analytique/symbolique, à base d'un simulateur ou à base de connaissances. De plus, lorsque c'est possible; le type de CA optimisé, le simulateur utilisé et l'implémentation (distribuée ou non) sont relatés. Enfin, les outils sont présentés dans un ordre chronologique d'apparition avec les auteurs et/ou les compagnies affiliées aux travaux.

Le Tableau 1.2 relate les outils commerciaux. Ce domaine compte, encore aujourd'hui, un nombre restreint de joueurs étant donné la nature complexe de la synthèse analogique qui se doit, dans un outil industriel, d'être toujours fiable et rapide.

Tableau 1.1 Liste des outils de synthèse analogique retrouvés dans la littérature

Nom de l'outil	Optimisation basée sur	Circuit	Catégorie	Simulateur	D.[12]	Année	Auteur affilié	Compagnie
AGO	Algorithme génétique (1.2.2.2)	tous	simulateur	HSpice [108]	non	2001	Langlois	LTRIM
ASF	Technique par descente de gradient et recherche stochastique (1.2.2.1)	tous	simulateur	Spectre	oui	2001	Krasnicki [55]	-
AMGIE	Stratégie de conception top-down par raffinement hiérarchique (utilise OPTIMAN [30])	tous	analytique	ISAAC [29]	non	2001	Van der Plas (Gielen)[84]	-
GSPICE	Algorithme génétique (1.2.2.2)	tous	simulateur	HSpice [108]	non	2000	Langlois	LTRIM
SADIC	Technique par conception d'expériences	tous	simulateur	*Spice*[13]	non	2000	Deval [21]	-
WiCkeD	Conception par centrage statistique	tous	simulateur	TITAN	oui	2000	Antreich [6]	Infineon
ANACONDA	Algorithme évolutionnaire avec nouvelle variante de recherche de patron (1.2.2.2)	tous	simulateur	TISpice[14]	oui	1999	Phelps (Krasnicki) [83]	Texas Instrument

[12] Distribution du calcul sur plusieurs ordinateurs mis en parallèles

[13] S'utilise essentiellement avec un simulateur électronique de la philosophie S.P.I.C.E.

[14] Texas Instruments Spice

Nom de l'outil	Optimisation basée sur	Circuit	Catégorie	Simulateur	D.[12]	Année	Auteur affilié	Compagnie
MAELSTROM	Nouvelle algorithme combinant génétique et recuit simulé (1.2.2.2, 1.2.2.4)	tous	simulateur	Spectre	oui	1999	Krasnicki [54]	
GPCAD	Programmation géométrique (1.2.1.1)	ampop	analytique	-	non	1998	Hershenson [38]	Barcelona Design
OPTOMEGA	Méthode du simplex et recuit simulé (1.2.2.1, 1.2.2.4)	tous	simulateur	Omega	non	1998	Keramat [43]	-
CONNAN	Recuit simulé amélioré (1.2.2.4)	tous	simulateur	HSpice [108]	non	1997	Marìn [64]	-
ASTRX/OBLX	Recuit simulé (1.2.2.4)	tous	simulateur	AWE[15]	non	1996	Ochotta [79]	-
KANSYS	Conception hiérarchique top-down sur base intensive de connaissance	-	connaissance	-	-	1996	Gupta [34]	-
FASY	Recuit simulé et sélection topologique par logique floue (1.2.2.4)	-	simulateur	Spice2 [71]	non	1996	Torralba [95]	-
JiffyTune	Formulation du Lagrangien adjoint avec optimisation par descente de gradient (1.2.2.1)	tous	simulateur	SPECS [75]	-	1996	Conn [15]	-
DARWIN	Algorithme génétique (1.2.2.2)		simulateur			1995	Kruiskamp [56]	
FPAD	Sélection topologique par logique floue	tous	analytique	-	-	1995	Mounir [70]	-

[15] Asymptotic Waveform Evaluator

Nom de l'outil	Optimisation basée sur	Circuit	Catégorie	Simulateur	D.[12]	Année	Auteur affilié	Compagnie
DORIC	Méthode de conception robuste (Tagushi)	tous	simulateur	HSpice [108]	non	1994	Daoud [18]	-
FRIDGE	Recuit simulé (1.2.2.4)	tous	simulateur	HSpice [108]	non	1994	Medeiro [69]	-
ARIADNE			connaissance			1993	Swings [94]	
ACAD	Simplification par réseaux 2 ports équivalents	plu-sieurs	symbolique	-	-	1993	Richman [86]	-
STAIC	Optimisation non-linéaire	biblio-thèque	symbolique	-	-	1992	Harvey [37]	Waterloo
UNICAD	Interface pour IDAC	-	-	-	-	1992	Goffart [32]	SGS Thomson
SEAS	Évolution simulée avec système expert (1.2.2.2)	-	simulateur	-	-	1991	Ning [76]	-
TOPICS	Conception hiérarchique avec assistance d'un système expert		connaissance			1991	Leenaerts [60]	
HECTOR	Sélecteur de topologie basé sur des règles heuristiques liées au domaine (utilise OPTIMAN)	biblio-thèque	connaissance	ISAAC [29]	-	1991	Swings [93]	-
OPASYN	Sélection de circuit par émondage heuristique	ampop	analytique	-	-	1990	Koh [45]	Berkeley
OPTIMAN	Recuit simulé (1.2.2.4)	tous	analytique	ISAAC[16]	-	1990	Gielen [30]	-

[16] Logiciel de simulateur électronique basé sur une représentation mathématique symbolique des performances du circuit évalué.

Nom de l'outil	Optimisation basée sur	Circuit	Catégorie	Simulateur	D.[12]	Année	Auteur affilié	Compagnie
OAC	Optimisation en 2 phases -Optimisation basée sur un modèle analytique -Optimisation numérique (LM) avec simulation SPICE (1.2.2.1)	ampop	analytique simulateur	*Spice*	-	1990	Odonera [80]	-
-	Optimisation analytique basée sur des règles liées à des cellules précaractérisées	tous	analytique	non	-	1990	Trontelj [96]	-
OASYS	Optimisation numérique limitée avec chasse du seuil (1.2.2.1)	biblio-thèque	connaissance	-	-	1989	Harjani [36]	CMU & Mentor Graphic
BLADES	Nouveau système expert & technique de conception formelle	ampop	connaissance	ADVICE	-	1989	El-Turky [24]	-
PRECISE-OPTIM	Descente par gradient (1.2.2.1)	tous	simulateur	PRECISE	-	1989	Dagonis [17]	-
DELIGHT-SPICE	Extension de la méthode des directions faisables avec contraintes fonctionnelles	tous	simulateur	Spice2 [71]	-	1988	Nye [77]	Harris Semi-conductor
ECSTASY	Algorithme convergent superlinéairement basé sur descente de gradient et recherche stochastique controlée (1.2.2.1, 1.2.2.4)	tous	simulateur	Spice3	non	1988	Shyu [92]	-
ADOPT	Optimisation non-linéaire (SUXES)	filtre	simulateur	*Spice*	-	1988	Lai [57]	-

Nom de l'outil	Optimisation basée sur	Circuit	Catégorie	Simulateur	D.[12]	Année	Auteur affilié	Compagnie
IDAC	-	-	connaissance	-	-	1987	Degrauwe [20]	SCEM
WATOPT	Optimisation non-linéaire de contraintes par utilisation de données du Jacobien (1.2.2.1)	-	-	-	-	1987	Chadha [14]	-
AIDE2	-	-	-	-	-	1985	Allen [3]	-
TILOS	Programmation posynomiale (1.2.1.1)	-	-	-	-	1985	Fishburn [25]	-
DELIGHT	-	-	simulateur	-	-	1981	Nye [78]	-

Tableau 1.2 Liste des outils commerciaux et industriels de synthèse analogique

Compagnie	Nom de l'outil	Circuit adressés	Algorithmes utilisés	OS[17]	S-B[18]	Optimisation adressés	Partenariat industriel
LTRIM Technologies Inc.	AGO *(interne)*	tous	Algorithmes génétiques (1.2.2.2)	WinNT	oui	Dimensionnement	-
Neo-Linear	NeoCircuit	tous	Nouveau recuit-génétique (1.2.2.2, 1.2.2.4)	Solaris	oui	Dimensionnement	Cadence
Analog Synthesis	AMS Genius	plusieurs	Intelligence artificielle et Fuzzy Art	Solaris & Linux	oui	Dimensionnement & Altérations Topologiques	Cadence & Synopsys
Barcelona Design	-Picasso -Dali -Gaudi -Miro	-Ampop -RF -SC Filter -PLL	Programmation géométrique (1.2.1.1)	Via le Web	non	Dimensionnement	Cadence
ComCAD ADS	Size!	tous	Modélisation mathématique	Solaris & Linux	non	Dimensionnement	-

[17] Système d'Exploitation

[18] Simulateur dans la Boucle

Les outils présentés se démarquent les uns des autres plus particulièrement au niveau de leurs algorithmes de recherche sophistiqués. La majorité d'entre-eux cherchent à tirer profit de l'hybridation de méthodes afin de favoriser les chances de succès des optimisations (section 1.2.3.1).

Dans le Tableau 1.1, on dénombre 4 outils utilisant spécifiquement des algorithmes évolutionnaires comme celui traité dans ce mémoire. Il s'agit là des outils ANACONDA[83], MAELSTROM[54], DARWIN[56] et SEAS[76].

Les outils utilisant des formes de recuit simulé sont : ASF[55], OPTOMEGA[43], CONNAN[64], FASY[95], FRIDGE[69], OPTIMAN[30]. Ces derniers sont généralement caractérisés par de long temps de calcul lors de leur exécution. Tout comme pour les outils SADIC[21], WiCkeD[6], DORIC[18] qui adoptent des méthodes de recherche statistique.

Enfin, il reste la gamme des outils qui utilisent une représentation analytique du domaine de recherche : AMGIE[84], GPCAD[38], TILOS[25]. Ceux-ci brillent indubitablement par leur grande vitesse d'exécution. Mais généralement, on y néglige les pertes de précisions dues aux erreurs sur la modélisation les accompagnant.

La majorité des autres outils d'optimisation sont ceux qui empruntent la voie des méthodes d'optimisation numérique : JiffyTune[15], STAIC[37], OAC[80], OASYS[36], PRECISE-OPTIM[17], ECSTACY[92], WATOPT[14] et ADOPT[57]. Bien sûr, comme expliqué à la section 1.2.2.1, ces méthodes sont accompagnées par un lot de limitations,

dont la plus importante, ces méthodes ne garantissent pas l'optimalité de la solution trouvée.

À la lumière des récents développements dans les outils de synthèse analogique, ce mémoire se classifie dans la catégorie des outils d'optimisation utilisant un algorithme génétique de type simple. L'état de l'art dans le domaine des AG se situe essentiellement au niveau de l'expérimentation d'opérateurs génétiques plus complexes (combinaison, sélection, mutation) que ceux énoncés à la section 1.2.2.2.1. De plus, les AG se prètent naturellement à la parallélisation. C'est pourquoi la majorité de ces outils sont de type *distribué*[19].

On observe aussi une certaine tendance à l'union des méthodes heuristiques avec les technologies de l'intelligence artificielle : KANSYS[34], ARIADNE[94], SEAS[76], TOPICS[60], HECTOR[93], BLADES[24] et IDAC[20]. C'est en soi une démarche logique quand on pense que les algorithmes heuristiques se substituent, l'intelligence en moins, au processus d'essai et d'erreur normalement effectué par l'humain. On comprend bien que l'étape suivante consiste à améliorer ces outils de synthèse avec une portion de raisonnement.

Dans les outils commerciaux, on remarque une tendance générale vers les outils d'optimisation paramétrique. Cependant, AMS Genius, grâce à son module d'intelligence

---

[19] Distribution du calcul sur plusieurs ordinateurs mis en parallèles

artificiel, va jusqu'à proposer des modifications topologiques au circuit lorsque les performances s'en trouvent améliorées.

Enfin, notons que certains outils sont limités à l'optimisation de circuits très spécifiques : *amplificateurs opérationnels*, *filtres* ou *bibliothèques de CA prédéfinies*. On le remarque généralement dans le cas où les méthodes employées ne font pas appel à un simulateur dans la boucle d'optimisation.

2 Matériels et méthodes

Dans ce chapitre, la description de l'outil d'optimisation sera abordée en situant d'abord le contexte de développement, les problèmes rencontrés et les solutions apportées. Une section portera spécialement le simulateur HSpice et ses caractéristiques particulières. Une révision des méthodes explorées afin d'améliorer la rapidité des simulations sera aussi couverte et une amélioration du fichier technologique sera présentée. Enfin, la structure à haut niveau du logiciel sera présentée ainsi qu'un survol des différentes consoles de contrôle. Une attention particulière sera portée au mécanisme d'évaluation de la qualité des solutions, *le module de pointage*, puisque celui-ci est, en un sens, l'élément clé de cet outil.

Bien sûr, pour des raisons de propriété intellectuelle appartenant à **LTRIM Technologies Inc.**, il est convenu que le code source (C++) du logiciel ne peut être directement relaté à l'intérieur de ce mémoire.

2.1 Historique de l'outil

L'outil d'optimisation a premièrement pris forme sous l'environnement UNIX dans les laboratoires du Groupe de Recherche en Microélectronique (GRM). La première version du logiciel était totalement dédiée par compilation à une optimisation très spécifique. Par manque de flexibilité et de versatilité relié à un outil compilé, de plus amples développements ont été envisagés. Un certain désir d'interactivité avec la déclaration de l'optimisation et la direction en cours de convergence a orienté le projet

vers la voie de l'intégration avec une interface graphique. Arrivé à ce point, la compagnie LTRIM Technologies s'est montrée intéressée à investir dans le perfectionnement de ce logiciel.

2.2 Logiciels et modules externes utilisés pour la programmation de AGO

Dans cette section, on présente les différents blocs logiciels utilisés pour bâtir l'outil d'optimisation nommé AGO[20].

2.2.1 Le compilateur (Borland C++ Builder v5.0)

Ce compilateur s'inscrit dans la philosophie de la programmation orientée objet avec plusieurs éléments visuels préprogrammés. Une vaste classe de bibliothèques (VCL[21]) permet au programmeur de construire un logiciel, en investissant peu d'énergie dans la programmation de l'interface graphique.

Un problème qui apparaît souvent avec la complexité grandissante d'un logiciel est la mauvaise gestion au niveau de l'allocation de la mémoire occasionnées par des oublis ou des erreurs de la part du programmeur. Dans « Builder v5.0 Professionnel », les pertes de mémoire sont détectées et identifiées par le module « CodeGuard ». Ce module alerte le programmeur lorsque des fuites de mémoire se produisent et il lui fournit suffisamment d'indications pour localiser et corriger la faute en question. Un exemple

---

[20] Analog Genetic Optimizer

[21] Virtual Components Library

classique concerne une fonction appelée dans le corps du programme à des millions de reprises et empreintes, à chaque appel, d'une petite fuite de mémoire. Ainsi à long terme, lors de l'exécution du logiciel, il peut résulter des pertes de mémoire vive très importantes, voir jusqu'à handicaper le fonctionnement du système d'exploitation. Étant donné la longue durée de fonctionnement du logiciel faisant l'objet de ce projet de recherche, les fuites de mémoire ont été énergiquement éradiquées des sections critiques constituant le cœur de l'optimisateur.

2.2.2 La bibliothèque graphique (ChartFX)

AGO devait permettre à l'utilisateur du logiciel de spécifier graphiquement les objectifs et les buts à atteindre par l'intermédiaire des courbes sur les performances du CA. Afin de permettre à l'utilisateur de voir et d'interagir avec les courbes de performances résultantes de la simulation du circuit, il fallait recourir à une interface graphique adaptée à ces besoins. Le groupe de composantes accompagnant le compilateur Builder C++ intégrait, en démonstration, une version simplifiée d'une composante d'affichage de diagramme. Après quelques recherches, la version commerciale complète (ChartFX v4.0) s'est avérée adéquate pour les besoins d'affichage des courbes. De plus, les fonctionnalités d'interactivité intégrées permettent à l'utilisateur du logiciel de pouvoir opérer graphiquement certaines configurations.

2.2.3 La bibliothèque des algorithmes génétiques (GALib)

La première bibliothèque utilisée dans ce projet fut la bibliothèque GALib programmée par Matthew Wall du MIT [101]. Elle est gratuite et elle fait aujourd'hui partie du domaine public. Elle encapsule plusieurs types d'algorithmes génétiques ainsi que plusieurs représentations génomiques pour s'adapter à différents types de problèmes d'optimisation. Pour nos besoins, nous nous sommes limités à l'utilisation d'un algorithme génétique dit stationnaire avec recouvrement de population (Steady-State), ainsi que d'une représentation génomique basé sur un vecteur avec des valeurs réservées en allèles.

La Figure 2.1 donne un bref aperçu des classes contenues dans la bibliothèque de GALib et utilisées dans le logiciel AGO.

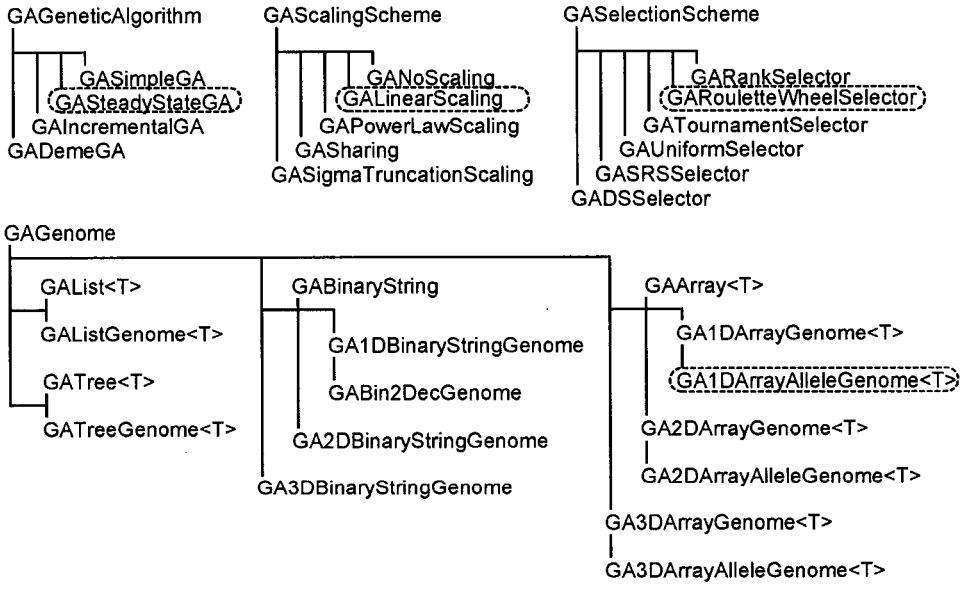


Figure 2.1 Classes de la bibliothèque GALib utilisée par AGO

On distigue à la Figure 2.1 que lors du choix de l'algorithme (GASteadyStateGA), il faut aussi spécifier le schéma de mise à l'échelle (GAScalingScheme) ainsi que le schéma de sélection (GaSelectionScheme). Dans notre cas, ceux-ci sont fixés à un échelonnage de type linéaire (GALinearScaling) et à un protocole de sélection des individus basé sur la roulette chanceuse (GARouletteWheelSelector). Le principe de la roulette chanceuse consiste à assigner à chaque individu une pointe de tarte sur une grande roue. La largeur de cette pointe est proportionnelle à la désirabilité de l'individu. Ainsi, lorsque la roulette est tournée pour effectuer un tirage, les probabilités de sélection favorisent les individus correspondant à une meilleure solution au problème.

Enfin, dans les classes de représentation génomique, c'est la classe GA1DArrayAlleleGenome qui fut sélectionnée. Ainsi, les paramètres sont alignés dans un vecteur unidimensionnel (1DArray) où les valeurs sont circonscrites à des ensembles finis (Allele). Cette classe fut choisie parce qu'elle était la mieux adaptée pour représenter en gènes les valeurs transmises en optimisation.

Afin de connaître de manière plus étendue les multiples fonctionnalités intégrées et offertes par cette bibliothèque, nous référons le lecteur à la documentation complète sur ce sujet [101].

2.2.4 La bibliothèque interpréteur d'équation (TbcParser)

Cet interpréteur d'équation, programmé en Pascal, est utilisé pour toutes les équations que l'utilisateur peut modifier à travers l'interface de AGO. Il définit une classe

(TbcParser) dont les données membres principales sont une équation, sous forme de chaîne de caractères, et une liste de paramètres que l'on peut fixer à une valeur scalaire afin d'évaluer le résultat de l'équation. La méthode *evaluate()* analyse la chaîne de caractères de façon à reconnaître l'équation qui s'y retrouve et évalue les résultats pour les valeurs de paramètre assignées. Les erreurs d'évaluation entraînent la création d'une exception de type *EbcParserError*.

2.3 Le simulateur de circuit (HSpice)

Le simulateur de circuit utilisé dans ce mémoire est nommé HSpice [108] et il est produit par la compagnie Synopsys[22]. Plusieurs raisons ont motivé ce choix. Premièrement, c'est une raison de disponibilité au GRM qui nous a initialement orienté vers ce choix. Deuxièmement, HSpice est un excellent simulateur qui possède une excellente stabilité de convergence au niveau de la résolution numérique. La réputation de HSpice, dans le milieu de la microélectronique, tient place de référence. Nous avons pu la comparer avec celle d'autres simulateurs commercialement disponibles comme T-Spice[111], SmartSpice[110] et PSpice[109]. Bien que semblable en vitesse de résolution, la fiabilité et la continuité des résultats générés en température ont toujours grandement favorisé l'utilisation de HSpice. Troisièmement, HSpice supporte une grande variété de modèles de composants électroniques en matière de transistors MOSFET et Bipolaires.

---

[22] Synopsys s'est porté acquisiteur de Avanti Corp. en 2001-2002

Comme nous le montrerons dans les sections qui suivent, le choix de HSpice comme simulateur de circuit pour réaliser l'évaluation des performances associées aux circuits ne s'est pas fait sans misère ni incertitudes.

Premièrement, la *version ASCII* des fichiers de résultats était d'une *précision limitée à 5 chiffres (0.00000E±00) et donc insuffisante*.

Deuxièmement, certains circuits produisent des *comportements non-désirables de la part du simulateur*; allant de l'impossibilité de converger jusqu'à l'arrêt impromptu de celui-ci.

Troisièmement, en première instance, il était très difficile de procéder, à l'intérieur d'*une même netlist*, à l'exécution de *plusieurs analyses différentes*.

Quatrièmement, les fonderies de puces ne divulguent normalement que les limites des procédés dans leurs fichiers technologiques. Ces coins extrêmes sont normalement utilisés par des concepteurs de circuits numériques. Par contre, pour émuler la véritable nature aléatoire des procédés à l'intérieur de ces bornes, il fallait une analyse de perturbations aléatoire reflétant de manière moins extrême les variations des procédés. Ainsi, l'application de *perturbations Monte Carlo* sur les composantes est devenue un volet important en association avec le simulateur. Une extension au fichier technologique a dû ainsi être développée.

2.3.1 Les fichiers de résultats de simulation de HSpice

On distingue 3 types de fichiers de sortie : le fichier principal en format ASCII, le fichier principal en format binaire et le fichier de mesure[23] en format ASCII.

2.3.1.1 Précision et fichiers ASCII

Les fichiers de résultats produits par le simulateur sont d'une *importance cruciale* pour le processus d'optimisation. Rapidement, le mode textuel (ASCII) de sauvegarde s'est avéré insuffisant pour une raison de précision. En effet, dans le cas de l'optimisation de certaines références de tension, une mantisse de 5 chiffres significatifs était nettement insuffisante (0.XXXXXE±YY) pour précisément évaluer les performances des circuits candidats. Ce problème était amplifié du fait qu'il n'existait aucune documentation publique sur le format de la sauvegarde des données dans le mode binaire.

2.3.1.2 Structure et fichiers binaires (.sw*, .ac*, .tr*)

Nous avons tenté d'obtenir de Avanti une documentation sur les formats binaires. Cette démarche s'est soldée par un refus catégorique de divulgation d'information interne à Avanti. L'option était claire, opter pour un autre simulateur ou faire de l'*ingénierie inverse* sur le *format binaire* de sauvegarde des fichiers de résultats de HSpice. Étant donné que la majorité des autres simulateurs de circuits sauvegardaient généralement uniquement en mode textuel, ils souffraient eux aussi d'un nombre de chiffres

---

[23] Fichiers complémentaires référant à des mesures effectuées sur des analyses (DC, AC, TR) et qui apparaissent lorsque les commandes `.MEAS` sont utilisées. Ces fichiers sont générés par le simulateur en post-traitement sur le fichier de simulation principal.

significatifs insuffisant (<6). En plus, ils ne partageaient pas la même fiabilité de convergence et de continuité en température propre à HSpice. L'analyse du fichier binaire de HSpice s'est encore une fois imposée d'elle-même. Le travail d'ingénierie inverse sur le format de fichiers binaires ne s'est pas fait sans difficultés. On présente à la Figure 2.2, la structure générale déduite pour un fichier typique.

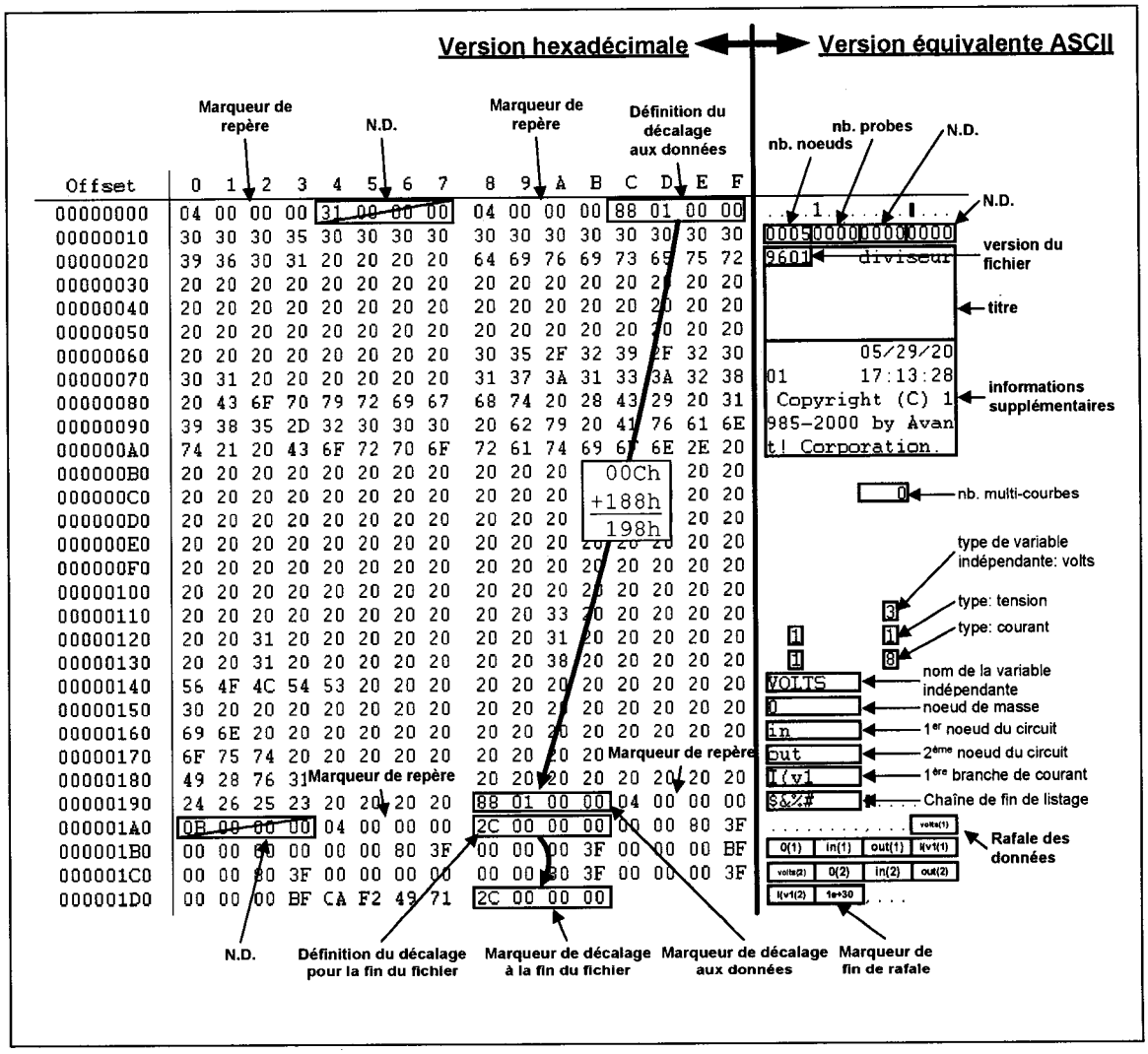


Figure 2.2 Structure et format des fichiers binaires HSpice

Le fichier débute avec une en-tête binaire de 4 segments de 32-bits. On y distingue des *marqueurs de repère* définis par les segments 04000000. Ces marqueurs servent essentiellement à séparer les secteurs du fichier. Chacun de ces séparateurs contient des informations utiles comme les valeurs de décalage exactes pour atteindre les secteurs suivants.

La trame suivante se compose aussi de 4 segments de 32-bits. Cette trame peut être segmentée en 4 valeurs numériques directement capturables en mode ASCII. Le premier membre représente le nombre de nœuds reportés dans le fichier. Cette valeur comptabilise aussi la variable indépendante balayée. Le second membre détermine le nombre de sondes[24] intégrées au fichier de sortie. Les sondes sont des mesures réalisées par le concepteur au niveau de la *netlist* et qui reporte dans le fichier des renseignements supplémentaires spécifiques au circuit. Enfin, les membres restants de cette trame demeurent non-identifiés et leur rôle n'a pas été déterminé en ce qui a trait à la lecture du fichier.

Le fichier contient ensuite deux gros blocs de données ASCII. Le premier membre du premier bloc détermine la version du fichier et donc, la structure utilisée pour emmagasiner les données. La version analysée suit le standard 9601. Nous savons que les formats 9007 et 2001 existent, mais nous n'assurerons pas ici leur compatibilité. Le reste du premier bloc reporte la première ligne de commentaire intégrée dans la *netlist*. Ce commentaire, figurant normalement en guise de titre, sera reporté jusqu'à concurrence de

---

[24] Provient de la traduction directe de "probes" en anglais.

60 caractères. Le second bloc reporte les informations sur la date de création, sur la compagnie Synopsys et sur ses droits d'auteurs.

Les premières informations sur les données emmagasinées commencent avec le nombre de tables présentes dans le fichier. Ensuite, une rafale de nombres est suivie par une rafale de noms. Les nombres représentent le type de valeur associée aux noms des nœuds. Par exemple, pour les nombres marqués d'un 1, il s'agit d'une valeur de tension. Pour les nombres marqués d'un 8, il s'agit d'un courant. Seule la première variable possède plusieurs alternatives, car elle décrit la variable indépendante de balayage. Comme dans l'exemple présenté, le nombre 3 représente un balayage en tension (VOLTS). Pour d'autres analyses, d'autres types de variables indépendantes apparaissent avec des nombres spécifiques et des noms significatifs tels que : HERTZ, TEMPERATURE, MONTECARLO, VOLTAGE, CURRENT, etc. La rafale des noms systématiquement intégrée dans le fichier se termine toujours par le marqueur $\boxed{\$\&\%\#}$. Dans la situation où des noms viennent à suivre ce marqueur, alors ils doivent être considérés comme des noms de sondes.

Enfin, le second segment *marqueur de repère* référé par le début du fichier apparaît juste après le dernier nom de nœud. Il découpe la section concernant les informations sur le fichier de celle concernant la *rafale des données*. Les données sont organisées de manière à correspondre avec l'ordre d'apparition des noms de nœuds présentés plus haut. Pour les fichiers DC et TR, les données sont de type réelles et tiennent sur 4 octets dans le format *IEEE valeur réelle à virgule flottante de type simple*

précision (*float*). Pour les fichiers AC, les données occupent le double d'espace, car le résultat de cette analyse s'exprime sous forme de nombres complexes et ils sont stockés avec une partie réelle et imaginaire (*2 floats*).

La fin d'une *rafale de données* est marquée par le nombre infini correspondant à 10^{30} ou CAF24571. La rafale de données peut être entrecoupée de plusieurs segments *marqueur de repère* lorsque le fichier devient très long (cas des simulations temporelles) ou lors de la présence de plusieurs tables. Enfin, le fichier termine toujours par une occurrence répétée du dernier *marqueur de décalage* clôturant la fin du fichier.

Il est fort possible que certaines propriétés avancées de ce format de fichier n'aient pas été adéquatement analysées ou correctement intégrées. Mais essentiellement, la capacité d'effectuer sans erreur les opérations qui suivent, démontre une couverture amplement satisfaisante pour nos besoins:

- Lecture d'une analyse temporelle contenue dans un fichier *très long*;
- Lecture d'une analyse en multi-courbe;
- Compréhension de tous les types d'abscisse (fréquence, température, paramètre, monte carlo, temps);
- Lecture des valeurs réelles et complexes.

En conclusion, bien que le travail d'analyse ait été relativement ardu, c'est surtout le travail de validation du module de lecture qui fut le plus pénible. En effet, il est toujours désastreux, une fois les fonctions de lecture terminées, de découvrir en cours

d'utilisation du logiciel qu'un aspect particulier de la structure du fichier n'avait pas été généralisé suffisamment pour faire face aux cas limites; non-envisagés.

Pour des raisons de sûreté, la capture des fichiers de simulation sous le format ASCII a aussi été intégrée dans le logiciel, malgré les limitations précédemment relatées. Quant à la durée de vie du format binaire utilisé, rien ne porte à croire que Synopsys vienne un jour à ne plus le supporter. En effet, malgré leur grande réticence à toute forme de divulgation, plusieurs autres outils de visualisation (extérieurs à Synopsys) restent dépendants de ces formats et doivent leur compatibilité aux normes adoptés par Avanti. C'est ce qui nous laisse croire que ces formats sont là pour rester.

2.3.1.3 Fichiers de mesures (.ms*, .ma*, .mt*)

Heureusement, les fichiers de mesures possédent un seul format de sauvegarde en mode ASCII avec une précision ajustable jusqu'à 10 chiffres significatifs par la commande `.OPTION MEASDGT=10`. Cette précision est amplement suffisante pour tous les besoins des optimisations.

2.3.2 Gestion des défaillances de HSpice

Il est possible, lors de l'optimisation, que certains circuits candidats proposés se retrouvent, du point de vue du simulateur, dans des régions non-réalisables de l'espace des solutions. Par conséquent, *trois* types de défaillances du simulateur peuvent survenir :

- Le simulateur avorte et le système d'exploitation détruit le processus; (*crash* ou *coredump*)

- Le simulateur se bloque et demeure figé pour une durée indéterminée; (*hang*)
- Le simulateur ne converge pas, mais termine de manière naturelle et reporte l'erreur de convergence rencontrée dans le listage de sortie; (*aborted*)

Afin de contrecarrer ces désagréments, un compteur de temps a été ajouté au logiciel (section 2.4.3.4). L'utilisateur peut ainsi programmer la période allouée à une simulation. Dépassé cette durée, le processus de simulation est terminé de manière abrupte et forcée. Cette manière de procéder permet d'éliminer les incertitudes liées aux deux premiers types de défaillances (*crash* et *hang*); les plus problématiques dans un processus d'optimisation ou des milliers de candidats doivent être simulés en séquence avec succès pour réussir l'optimisation désirée.

2.3.3 Considérations sur la vitesse d'exécution de HSpice

Une considération de grande importance, dans le contexte d'optimisations utilisant un simulateur dans la boucle, est d'optimiser le temps utilisé par chacune des simulations. Plusieurs stratégies sont présentées dans cette section afin de maximiser l'information retirée des circuits et minimiser le temps de calcul qui lui est nécessairement associé.

2.3.3.1 Vitesse par le nombre de points dans les analyses

Une méthode directement employée pour accélérer le temps d'exécution de la simulation concerne la réduction du nombre de points dans les analyses. Ainsi, il est

possible de trouver un compromis acceptable entre la réduction du temps de calcul et la résolution des analyses pour les fins d'optimisation.

Par exemple, l'analyse de la sensibilité aux fluctuations de l'alimentation (PSRR) d'une référence de tension par rapport à une variation de la température se réalisera normalement par 1 à 2 points par décade pour un balayage fréquentiel allant de 0.01 Hz jusqu'à 10 GHz. La température sera balayée en 5 à 7 points allant de; –40 °C à 85. °C (plage industrielle).

En tout, l'évaluation sera composée de **12(décades)x2(points/décade)x5(points en T°) = 120 points** (Figure 2.3a). Ceci comporte beaucoup moins de points que la même analyse utilisée pour valider, sur tout son spectre, le bon fonctionnement du circuit qui peut contenir plus de **12(décades)x21(points/décade)x21(points en T°) =5292 points** (Figure 2.3b). Étant donné que certaines analyses n'ont pas nécessairement besoin d'un balayage uniforme ou exhaustif, il est utile dans ce cas d'utiliser la déclaration POI qui permet de dicter spécifiquement point par point les endroits où auront lieu les analyses (Figure 2.3c).

```
.AC POI 6 0.1 10k 100k 1x 10x 100x
```

Listage 2.1 Exemple d'analyse utilisant un balayage par points spécifiques.

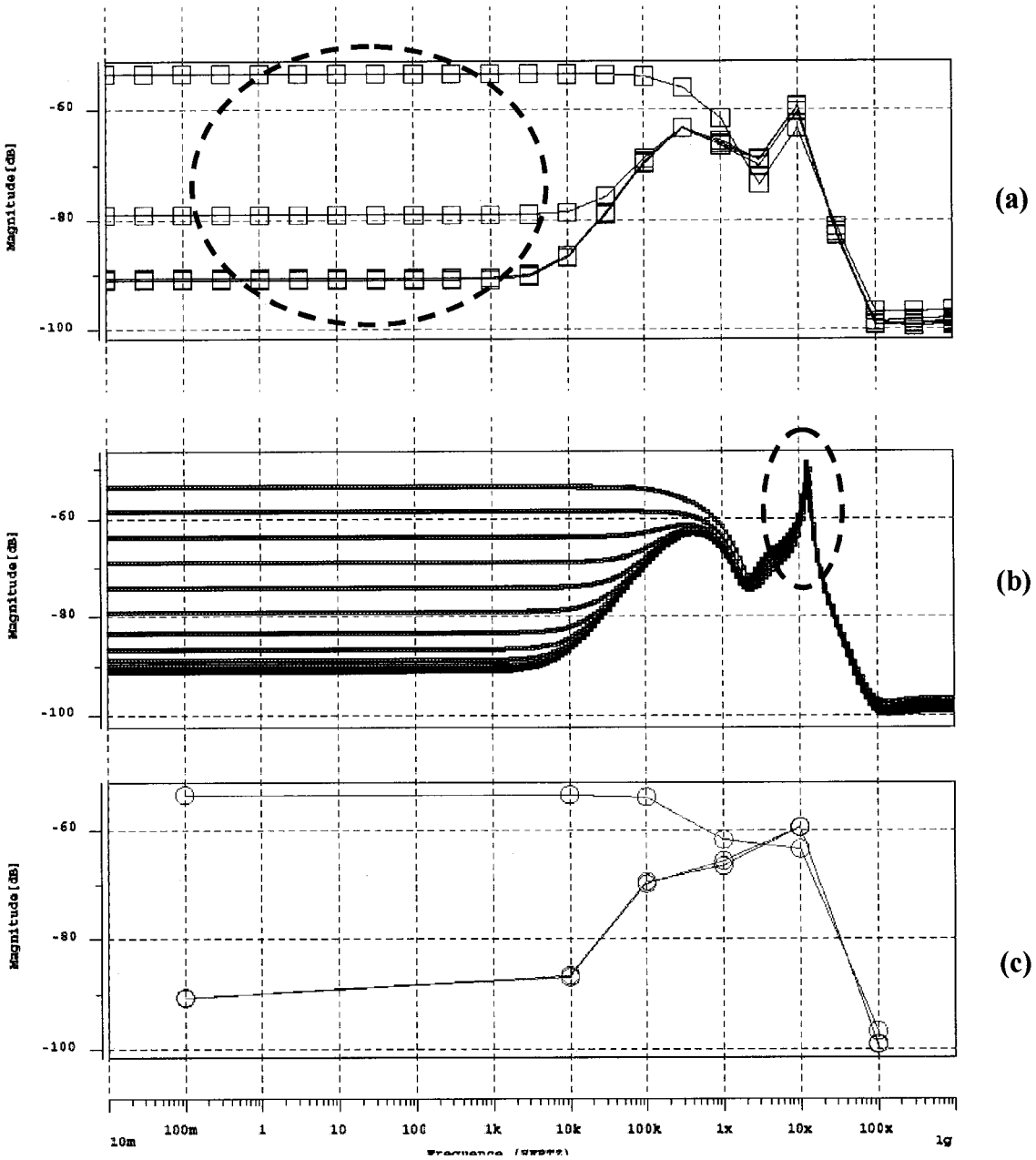


Figure 2.3 Comparatif sur la résolution de l'analyse du PSRR : (a) analyse en cours d'optimisation, (b) analyse en validation, (c) analyse avec un nombre réduit de points.

Par exemple, pour la même analyse en fréquence du PSRR, une analyse contenant 60 points contient beaucoup de redondances en ce qui concerne les valeurs à basse

fréquence (voir encerclé à la Figure 2.3a). En effet, les points situés entre 0.1 Hz et 10 kHz sont essentiellement identiques et représentent la bonne fonctionnalité du PSRR. À haute fréquence, la performance se dégrade plus significativement et son allure fournit généralement des informations vitales sur les possibilités d'instabilité indésirables pouvant apparaître ensuite lors de l'analyse transitoire (section 2.4.4.4).

Il est ainsi possible de limiter l'analyse fréquentielle à 1 point à basses fréquences et 5 points à haute fréquence. Il est bien important de bien situer les points directeurs afin d'éviter l'apparition de phénomènes d'amplifications entretenues au sein de la référence de tension. Le bilan de l'évaluation peut ainsi être réduit à **6(points Hz)x3(points T°)= 18 points**. Il est bien important de correctement positionner les fréquences auxquelles seront échantillonnées le comportement du circuit, car au cours de son optimisation, si d'excellentes performances relatées par les points discrets de l'analyse se fait au détriment de performances sur des portions non-couvertes du spectre, c'est alors un circuit handicapé qui prendra la tête.

En résumé, les cas de fuite ou de débordement requièrent généralement une vigilance continue de la part de l'utilisateur de l'outil d'optimisation.

2.3.3.2 Gain en vitesse par les options

L'accélération de la vitesse des simulations peut être obtenue par l'activation de judicieuses *options* propres à HSpice par la commande `.OPTIONS`. Voici une liste des options couramment utilisées :

Tableau 2.1 Listes de certaines options utilisées pour accélerer la simulation

ICSWEEP=1	se base sur un calcul précédent pour initialiser le suivant
ITL5=X	limite une analyse transitoire à un maximum de X points
NOTOP	ne réalise pas la procédure de vérification d'intégrité topologique du circuit
AUTOSTOP=1	arrête une simulation transitoire aussitôt que toutes les mesures ont été réalisées
PROBE	élimine de la sortie les nœuds intégrés par défauts
POST=1	sauvegarde en format binaire; moins volumineux; plus rapide au chargement
PIVOT=13	maximise le temps et les ressources mémoires lors de la résolution numérique des matrices

Chacune de ces options permet de sauver un peu de temps de calcul, mais elles doivent être utilisées avec beaucoup de précaution et en juste connaissance de cause. Par exemple, certaines de ces options, dans une phase de développement du circuit, peuvent se révélées fatales pour la convergence du circuit.

2.3.3.3 Gain en vitesse par concaténation de circuits

Une des stratégies investiguées, pour accélérer l'évaluation des circuits candidats, consistait à concaténer à l'intérieur d'une même *netlist* plusieurs descriptions du même circuit. Par exemple, pour évaluer 10 circuits candidats, une *netlist* peut être créée avec les descriptions répliquées de 10 fois le même circuit; en s'assurant, bien sûr, qu'il n'y ait aucun chevauchement de nom de composants, de nœuds ou de paramètres à optimiser.

Cette expérience s'est révélée fructueuse, mais seulement pour des cas très particuliers. En effet, la concaténation en un bloc de plusieurs circuits à tester engendre un grand risque que si l'un des circuits ne connaît pas de convergence, alors les autres circuits analysés en même temps ne pourront pas être évalués puisque que le simulateur avortera la simulation.

2.3.3.4 Gain en vitesse par concaténation en .DATA

Une stratégie alternative à celle présentée à la section précédente consiste à utiliser la commande .DATA afin de balayer en rafale les différents circuits candidats dans la même *netlist*. La commande .DATA permet de substituer, itérativement, jusqu'à un maximum de 32 paramètres simultanés dans une *netlist*.

De cette manière, contrairement à la solution précédente, un seul circuit est simulé à chaque capture des paramètres dans le fichier de données. Ce fichier de données est en fait construit à partir des paramètres à optimiser proposés par l'AG.

Il subsiste cependant toujours le même problème que pour la solution précédente en ce qui concerne les éventualités de défaillances. En effet, la simulation d'un groupe de candidats du circuit peut être interrompue aussitôt qu'un circuit rencontré se trouve à ne pas converger. Éliminant, par conséquent, l'évaluation de tous les autres circuits qui suivent. Aucun remède jusqu'à présent n'a été trouvé, à travers les options du simulateur, pour pallier à ce problème. De plus, advenant l'utilisation de cette stratégie, la gestion des

cas avortés resterait une tâche relativement compliquée à implémenter au niveau logiciel. Mais cette solution subsiste quand même dans le domaine de l'envisageable.

Entre autre, à travers l'évaluation sommaire des méthodes d'accélération de la procédure de simulation, le gain en vitesse ne s'est pas montré aussi substantiel qu'escompté, comparé aux stratégies expliquées aux sections 2.3.3.1, 2.3.3.2 et 2.3.3.6.

2.3.3.5 Gain en vitesse par avortement prématuré

Cette stratégie consisterait, à travers une analyse critique et effectuée en première instance, à déterminer la viabilité du circuit en évaluation et ainsi permettre l'avortement prématurément de la simulation si celui-ci ne rencontre pas un certain critère de passage. Malheureusement, les versions courantes de HSpice n'offrent aucune *instruction de contrôle* pouvant permettre l'intégration de ce type de stratégie pour accélérer le processus d'optimisation.

2.3.3.6 Gain en vitesse par réduction du nombre de nœud

Une des portions significative du temps pris par la simulation, à l'exception du calcul lui-même, concerne la sauvegarde et la lecture des fichiers de résultats. En effet, lorsque l'option `POST` est activée, le simulateur sauvegarde par défaut toutes les tensions des nœuds ainsi que tous les courants délivrés par les sources de tension. Cette sauvegarde n'est cependant aucunement problématique lorsque le circuit comporte relativement peu de nœuds. Cependant, pour certaines longues simulations, comme celles dans le domaine temporel, il est possible de générer d'immenses fichiers de résultats

(parfois de plusieurs centaines de kilo-octets). Or, la sauvegarde d'un *nombre réduit de données* peut se répercuter comme un gain en temps. Au même titre, l'opération de lecture effectuée par la suite sur ce fichier s'en trouvera aussi accélérée.

Il est important de porter une attention particulière au fonctionnement de la simulation en transitoire. En effet, il est possible qu'une simulation cherche à affiner son pas d'incrément temporel durant la simulation pour des questions de continuité. Cette adaptation du pas d'intégration augmente de manière considérable le nombre de points à sauvegarder. Il est donc possible, à travers l'option `INTERP=1`, de forcer les résultats de sortie a obéir à un interval temporel fixe. De cette manière, la courbe de sortie ressemblera à des segments brisés, mais leur précision ne pourra pas être mise en cause. Le seul danger pouvant subsister concerne les mesures réalisées par des interpolations sur ces segments. C'est pourquoi, il est fortement déconseillé d'utiliser des mesures `.MEAS` en conjonction avec l'option `INTERP` activée.

Certaines situations désagréables peuvent se produire lorsque le circuit adopte une certaine propension à osciller. En effet, le nombre de points nécessaires à la description temporelle des tensions des nœuds augmentera très rapidement car le simulateur ne parviendra pas à relaxer le pas d'intégration. Or, dans cette situation, la taille du fichier de sortie peut littéralement exploser. Afin d'éviter ce cas particulier, l'option `ITL5` devient d'une grande aide. De fait, cette option a pour rôle de limiter le nombre de points à la sauvegarde pour une analyse temporelle. Par exemple, si `ITL5=5000` alors la simulation temporelle ne pourra sauvegarder plus de 5000 points dans le fichier de sortie et avortera

ensuite. Pour des situations normales, une simulation de *vérification du démarrage automatique* d'une référence de tension doit, par exemple, se tenir entre 100 et 2000 points. Ainsi, un cas de débordement à 5000 points démontre déjà, en un sens, une mauvaise fonctionnalité qui devrait conduire au rejet fondé du circuit candidat.

2.3.4 Concaténation des analyses dans une seule *netlist*

Le concepteur désire généralement pouvoir effectuer un certain nombre d'analyses différentes sur son circuit. Il s'attend, pour ce faire, à ce que le simulateur lui offre une gamme d'instructions de contrôle lui permettant de changer, de manière automatisée, les analyses auxquelles il soumet son circuit et de s'adapter aux différentes configurations de connexions associées à chacune de ces analyses. Un des opérateurs populairement relaté dans la documentation est l'instruction `.ALTER`. Hélas, cette instruction ne s'exécute pas de la manière attendue lorsqu'elle est utilisée. Bien au contraire, son comportement est même contre-intuitif. De manière symptomatique, elle génère toujours plus d'analyses qu'escompté. Ce comportement indésirable ne peut être résolu qu'après avoir sérieusement analysé et cerné le rôle de cette instruction. C'est pourquoi, il est important de passer en revue les instructions allouées lors de l'utilisation de l'instruction `.ALTER` :

- Redéfinition sur un paramètre (`.PARAM`)
- Redéfinition sur un composant
  - Changer les noms des nœuds
  - Changer les valeurs passées en paramètre à ce composant

- Ajouter une analyse (.DC, .AC, .TR)
- Inclure ou retirer une bibliothèque (.LIB, .DELLIB) pour :
  - Changer de composants technologiques
  - Positionner les composants à un coin extrême de la technologie

Grâce à l'utilisation des instructions d'ajout/retrait de bibliothèques (.LIB et .DEL LIB), il est possible d'éliminer les analyses redondantes et impertinentes. Cette astuce *n'est pas explicitement documentée au chapitre 3 du manuel utilisateur de HSpice* [108] de la manière qu'elle est présentée ici. De même, il est possible que cette présentation ne soit fonctionnelle que par un concours de circonstances. En effet, ces instructions ont été initialement intégrées dans cet outil SPICE pour la substitution automatisée de différents fichiers technologiques liés aux composantes électroniques et non pas aux analyses multiples.

Afin d'obtenir une seule *netlist* avec des analyses multiples, il faut se conformer à la solution suivante qui consiste à encapsuler les instructions modificatrices du circuit à l'intérieur de bibliothèques (.LIB, .ENDL).

Le Listage 2.2 présente une utilisation de l'instruction .ALTER qui débouche sur la production de 3 analyses distinctes (DC, AC et TR), effectuées respectivement selon leurs topologies propres de connexion. Cette voie permet donc une meilleure structuration et simplifie la réutilisation des analyses associées à un type de circuit particulier.

```
*Netlist présentant comment utiliser adéquatement les .ALTER
$le circuit
R1 1 0 R=1k
V1 1 0 DC=1

$les analyses
.LIB 'fichier_courant.sp' DC
.ALTER
.DEL LIB 'fichier_courant.sp' DC

.LIB 'fichier_courant.sp' AC
.ALTER
.DEL LIB 'fichier_courant.sp' AC

.LIB 'fichier_courant.sp' TR
.END

$les bibliothèques
.LIB DC
.DC V1 0 1 0.1
.ENDL DC

.LIB AC
R1 2 0 R=1m AC=1k
V1 2 0 DC=0 AC=1
.AC DEC 5 1 10
.ENDL AC

.LIB TR
R1 2 1 R=1m AC=1k
C1 1 0 C=10uF
V1 2 0 PWL(0us,0V 1us,0V 2us,1V)
.TRAN 1u 10u
.ENDL TR
```

Listage 2.2 Exemple de *netlist* présentant une utilisation conforme de l'instruction `.ALTER`.

De plus, cette méthodologie retire toute forme de restriction quant au nombre d'analyses pouvant être effectuées en séquence. Enfin, en opposition à la solution alternative, qui consisterait à distribuer dans des fichiers séparés les différentes analyses,

cette stratégie ne nécessite qu'*un seul appel* à la ligne de commande lors du lancement de la simulation.

2.3.5 Définition Monte Carlo pour les analyses

L'analyse Monte Carlo est une facette très importante dans la caractérisation des CA. Le but de l'analyse Monte Carlo est d'assujettir les différentes composantes du circuit à des variations physiques aléatoires, comme celles qui se produisent réellement lors de l'étape de fabrication. Or, la fonderie fournit normalement aux concepteurs de circuit, seulement les coins extrêmes des procédés de fabrication. Ces coins offrent une marge de manœuvre suffisante à la fonderie et sensibilise le concepteur aux pires ou meilleurs cas de fonctionnement pouvant affecter son circuit. Le concepteur est ainsi chargé de vérifier si son circuit est en mesure de supporter de tels écarts paramétriques.

Lorsqu'un CA est robuste aux variations des procédés, cela signifie que les fonctionnalités ou performances en sortie du circuit se préserveront dans une certaine mesure autour des valeurs prédites. La robustesse des CA face aux variations des procédés est en soi très difficile à formuler et à rencontrer. D'autant plus que les limites extrêmes fournies sont généralement utilisées pour valider des designs numériques. C'est pourquoi, elles ont généralement, du côté analogique, la réputation de littéralement *casser* les circuits. Pour cette raison, les coins extrêmes ne sont pas conseillés pour l'évaluation de la robustesse des CA face aux variations des procédés.

En remède à cette problématique, une bibliothèque supplémentaire a été créée afin de mieux modéliser des perturbations aléatoires plus sujettes à se produire dans la pratique. Bien sûr, les coins extrêmes sont utilisés comme des *points de repère* à ces nouveaux modèles de fluctuations aléatoires sur les paramètres des composants.

Les sections de la bibliothèque dont le suffixe est '**_ABS**' affectent les variations aléatoires communes à un type de composant. Cette variation peut, par exemple, aller jusqu'à atteindre un des coins du procédé lorsque la variable atteint le $\pm 3\sigma$ de la distribution gaussienne aléatoire associée. On assigne la notion de *variation absolue* à cette perturbation affectant de manière commune un ensemble de composants.

Les sections de la bibliothèque référées au suffixe '**_PARAM**' entreposent les paramètres technologiques dirigeant les variations absolues et relatives. On y retrouve généralement la valeur nominale, la valeur de la déviation (*absolue*) et valeur de la dispersion (*relative*).

L'organisation du fichier technologique est relativement simple lorsqu'elle est mise sous forme schématisée (Figure 2.4). Les flèches marquées d'un '**1**' concernent le chargement du fichier technologique original de la fonderie (TSMC) en spécifiant le coin du procédé désiré :

- **TT** : Procédé typique où les NMOS et les PMOS sont au centre du procédé en fait de rapidité de commutation et de valeur de la tension de seuil.
- **SS** : Procédé où les NMOS et les PMOS sont lents.

- **FF** : Procédé où les NMOS et les PMOS sont rapides.
- **SF** : Procédé où les NMOS sont lents et les PMOS sont rapides.
- **FS** : Procédé où les NMOS sont rapides et les PMOS sont lents.

Les flèches marquées d'un '**2**' présentent les bibliothèques chargées dans le simulateur pour intégrer les variations Monte Carlo (*absolues* et *relatives*) à l'aide de véritables variables aléatoires :

- **MC_REEL** : Procédé à variations aléatoires centré autour de la valeur nominale typique.

Cette structure des fichiers technologiques a été privilégiée parce qu'elle permet un maximum de versatilité vis-à-vis les *netlists* des circuits à simuler. Que la *netlist* soit constituée de déclarations en composantes (M, D, Q, R) ou d'appel aux sous-circuits équivalents (xM, xD, xQ, xR), l'appel à la bibliothèque 'mm0355.l' ou 'lt035tsmc.lib' reste toujours valide, compatible, transparente et sans modification supplémentaire à apporter à la *netlist*.

Lorsque le concepteur fait appel à la bibliothèque 'mm0355.l', il est libre de charger seulement les bibliothèques qui lui semblent nécessaires (transistor, diodes, résistances, bipolaires). Cependant, pour la bibliothèque 'lt035tsmc.lib', cette liberté n'est plus permise au concepteur (à moins que celui-ci ne court-circuite volontairement le processus).

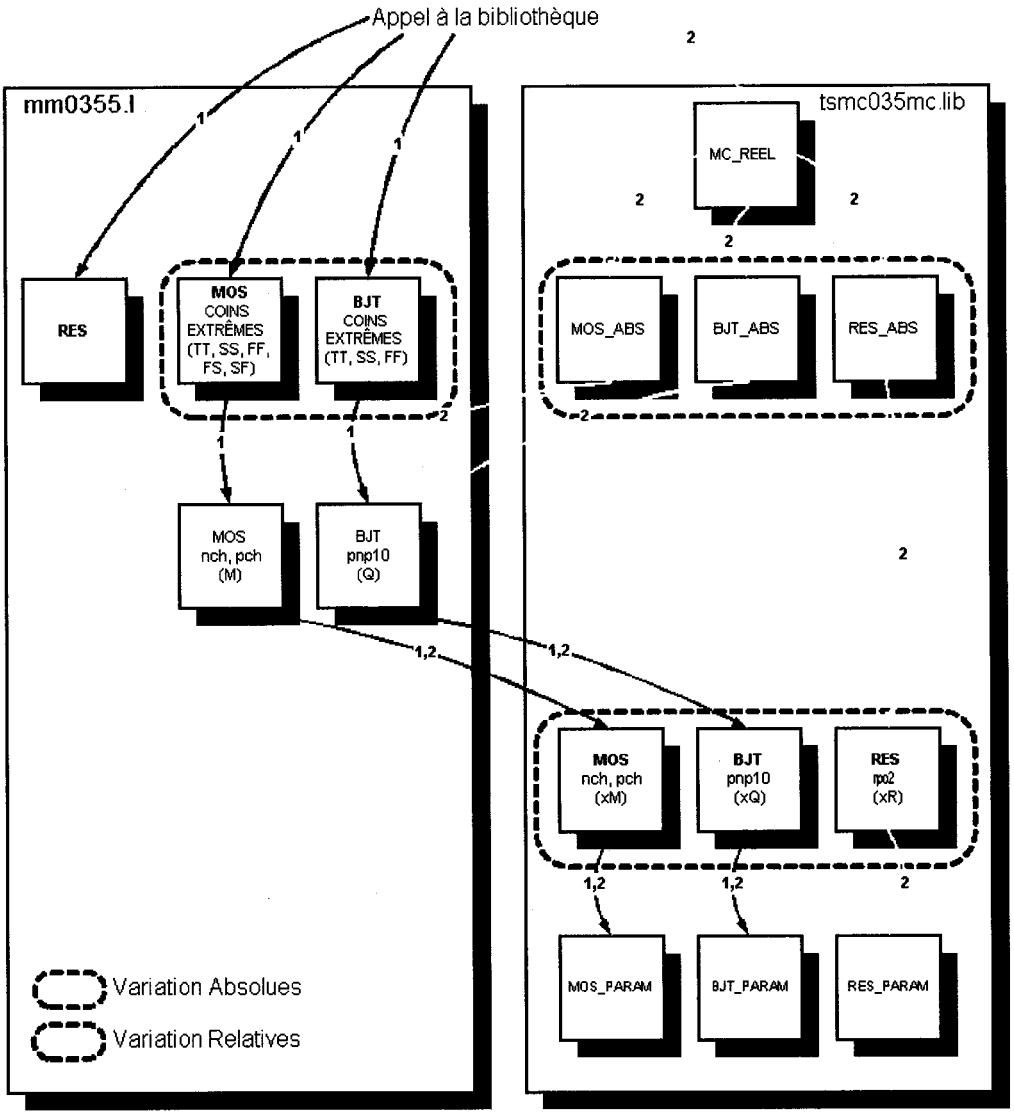


Figure 2.4 Diagramme de flot du fichier technologique adapté aux analyses Monte Carlo

Le listage du fichier 'lt035tsmc.lib' est référé en Annexe I (Listage 4.3). Le fichier 'mm0355.l' étant très volumineux et redondant, seulement les sections ayant subies une modification sont relatées.

2.4 Structure du logiciel AGO

La section suivante présente la structure du logiciel AGO. On couvre d'abord la hiérarchie des classes et on présente les diagrammes de fonctionnement de celui-ci. Enfin, on termine par un survol des différentes interfaces du logiciel avec une brève description de leurs rôles. Une attention particulière est apportée au *module de pointage* qui fait l'objet de la section 2.4.4.

2.4.1 Hiérarchie des classes utilisées dans AGO

La hiérarchie des classes présente la structure au niveau intérieur du logiciel à la Figure 2.5. Ces classes forment le maillage des différents modules sous-jacents au logiciel AGO et référé par l'AG et les différentes consoles de contrôle accessibles à l'utilisateur.

La *classe Optimisation* est la classe racine au logiciel. On remarque que la *classe Circuit* contient 3 *netlists* : originale, explosée et optimisée. La raison de l'existence de la *netlist* explosée provient de considérations au niveau de la simulation du circuit dans un répertoire temporaire et la nécessité de référer dans la *netlist* à des bibliothèques ou des fichiers d'inclusion externe, ce qui implique nécessairement la création d'un fichier qui concatène tous ces fichiers pertinents. Enfin, lorsqu'un appel à une classe se termine par des crochets ([ ]), cela signifie que plusieurs instances de cette classe peuvent être déclarés. Ce qui est généralement le cas pour la classe des *Pointeur* qui doit dicter tous les objectifs spécifiques pour chacune des performances.

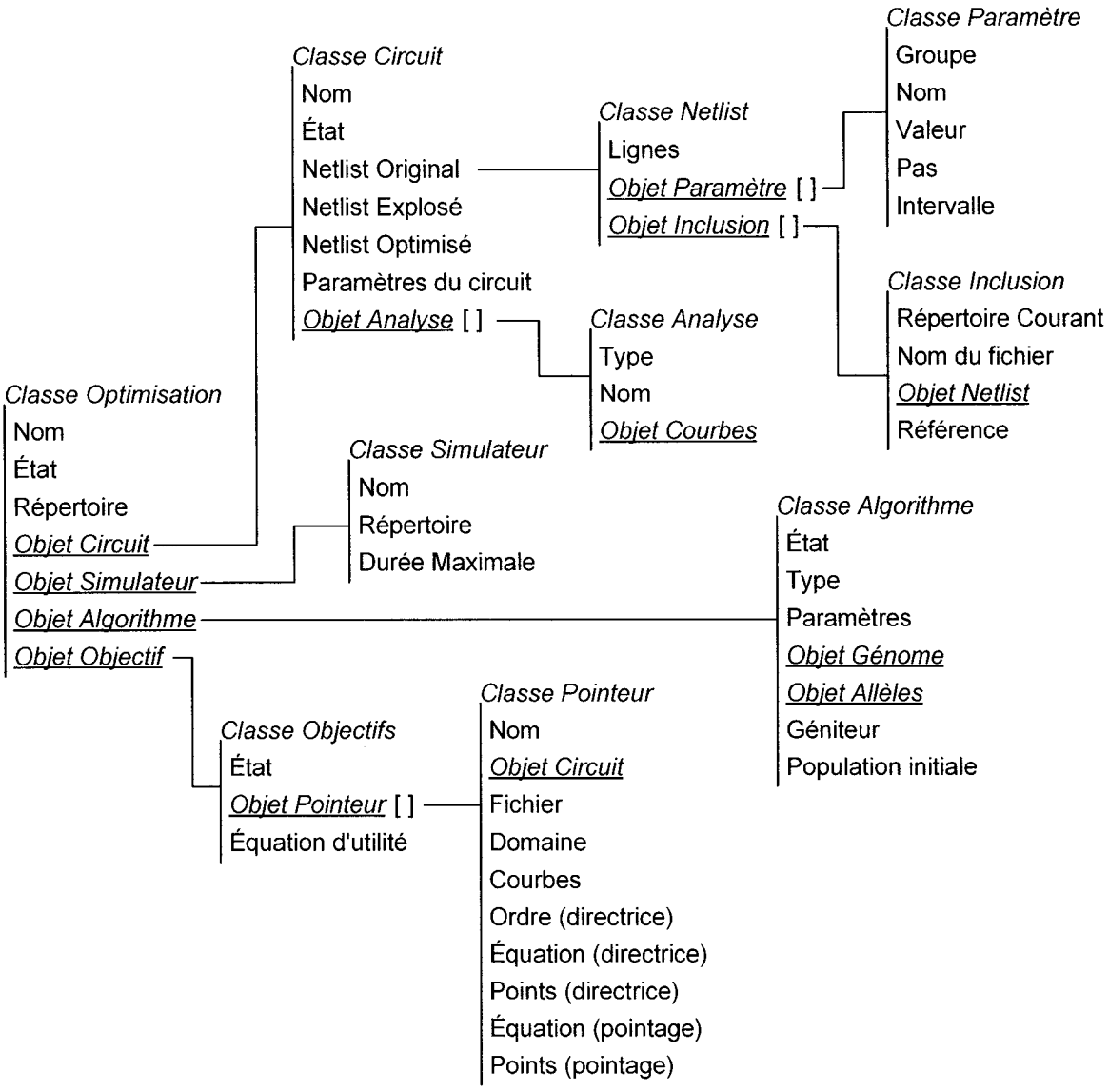


Figure 2.5 Hiérarchie des classes utilisées dans le logiciel d'optimisation AGO.

2.4.2 Diagramme de flot sur le fonctionnement de AGO

Cette section présente la structure algorithmique du moteur d'optimisation définit dans le logiciel AGO.

Figure 2.6 Principe du logiciel d'optimisation AGO.

Afin que l'AG puisse opérer une optimisation sur un CA, il faut procéder à plusieurs traitements des informations venant des descriptions des CA pour les rendre conforme aux requis du moteur d'optimisation. Pour ce faire, la première étape consiste à charger, dans le logiciel, la description textuelle du circuit (Figure 2.8) et ensuite de procéder au filtrage des paramètres entrant dans l'optimisation (section 3.2).

Lorsque le moteur d'optimisation est démarré (Figure 2.7), les différentes configurations ajustées par l'utilisateur sont prises en compte: fichier de résultats à charger, détermination de la représentation génique des paramètres à optimiser, le type d'AG sélectionné, etc.

Par la suite, l'algorithme crée une population initiale au hasard selon le nombre d'individus spécifiés dans les paramètres de contrôle de l'AG. Facultativement, le candidat nominal venant du fichier original sera intégré à cette population initiale. Le bon côté c'est qu'il est possible d'intégrer dès le départ une solution sous-optimale, autour de

laquelle l'algorithme va explorer. Le mauvais côté réside dans le fait qu'avec une petite population, cette solution sous-optimale risque de piéger rapidement l'optimisation à ce minimum local. Un remède simple palliant à ce problème réside normalement dans la maintenance d'une certaine diversité génétique; qui est normalement favorisée lors de l'utilisation de populations nombreuses.

Le bloc ombragé du diagramme fonctionnel de la Figure 2.7 concerne le fonctionnement interne de l'AG. En effet, la bibliothèque des AG utilisée nécessite une certaine initialisation afin que soient configurés certains de ses paramètres de contrôle. La déclaration d'une *fonction d'objectif*, que l'on voit dans le diagramme représenté par le *bloc évaluateur du circuit*, doit aussi être définie pour que l'algorithme puisse opérer correctement. Comme pour tous les algorithmes heuristiques, ce bloc constitue, pour l'engin d'optimisation, le seul moyen d'obtenir une correspondance entre *un gène proposé* à l'évaluation et *la valeur de performance qui lui est attribuable*.

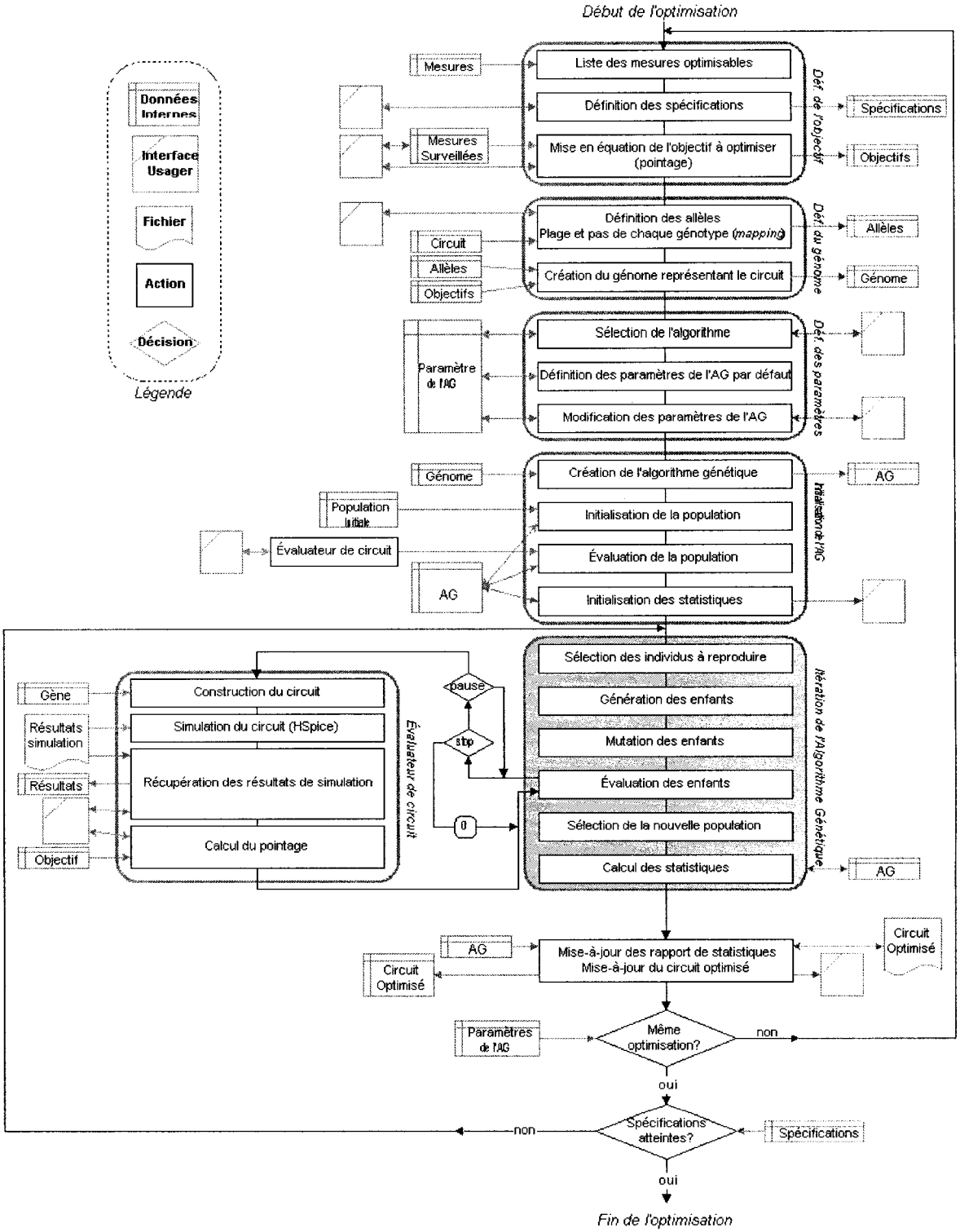


Figure 2.7 Diagramme fonctionnel du moteur d'optimisation du logiciel AGO

2.4.3 L'interface graphique de AGO

L'interface du logiciel est séparée en trois sections principales : le processus général de fonctionnement du logiciel, les consoles de contrôle ou de modification des paramètres et les fenêtres de visualisation des statistiques ou des résultats en cours d'optimisation.

Le processus général de fonctionnement de AGO est présenté à la Figure 2.8. On remarque que le moteur d'optimisation est lancé comme un processus indépendant (*thread*), afin de ne pas bloquer l'interface graphique de l'outil lorsque celle-ci est en train d'effectuer l'optimisation d'un circuit. Bien sûr, le résultat optimisé est rafraîchi de manière continue, à l'extérieur du logiciel, dans un fichier prévu à cet effet.

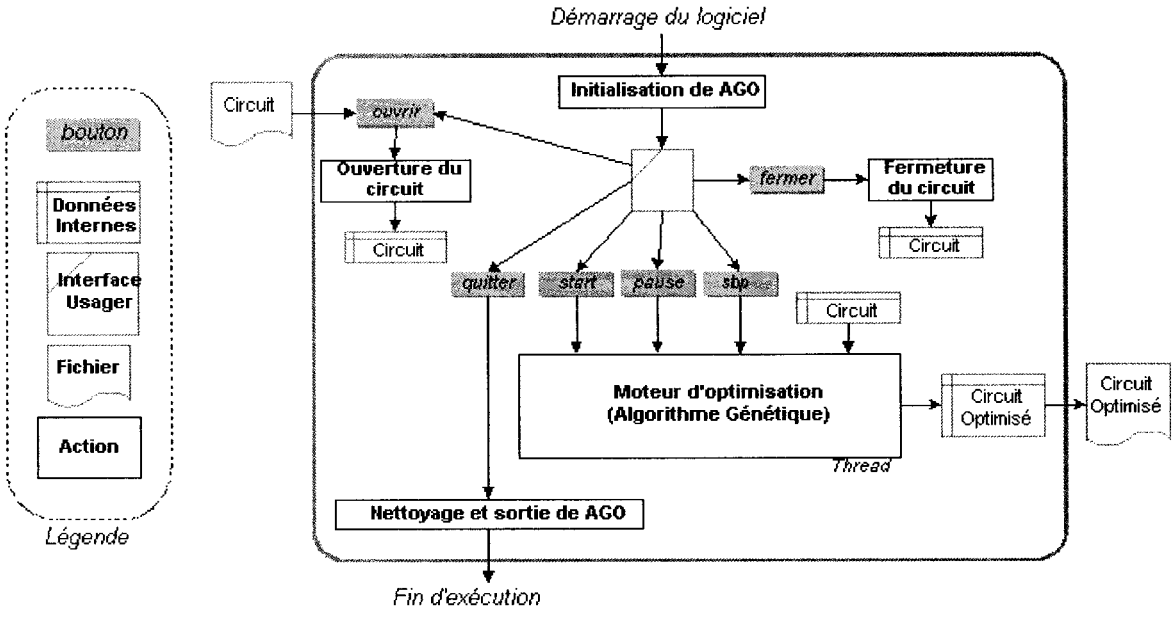


Figure 2.8 Structure fonctionnelle et interactive de l'interface du logiciel d'optimisation AGO

2.4.3.1 Console des paramètres de contrôle de l'algorithme génétique

Cette console permet à l'utilisateur de régler les différents paramètres associés à l'AG (Figure 2.9). On trouve à la section 3.4 une description détaillée des paramètres importants à spécifier. Notons que pour réaliser des optimisations de nature différente, il est nécessaire de changer la valeur du *Seed* (germe initialisateur du générateur de nombres aléatoires).

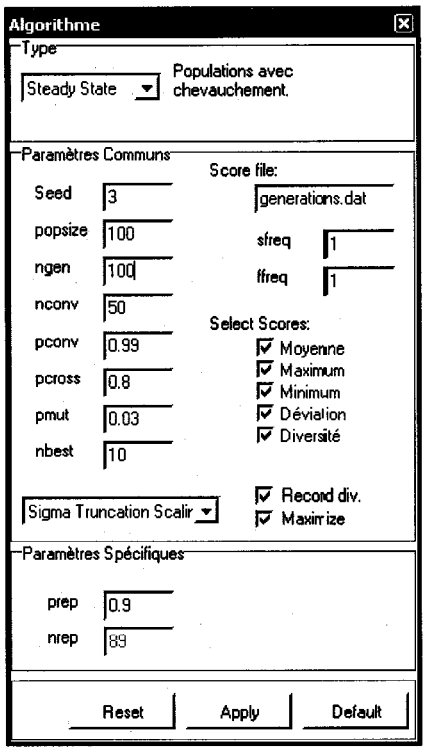


Figure 2.9 Console de configuration des paramètre de contrôle de l'algorithme génétique

2.4.3.2 Console d'affichage des paramètres sous optimisation

La console d'affichage présente les paramètres optimisables capturés dans la *netlist* originale (Figure 2.10). En plus des valeurs de paramètres du candidat *courant* et

du *meilleur* candidat, cette console présente aussi les bornes limites et le pas, propres à chacun de ces paramètres.

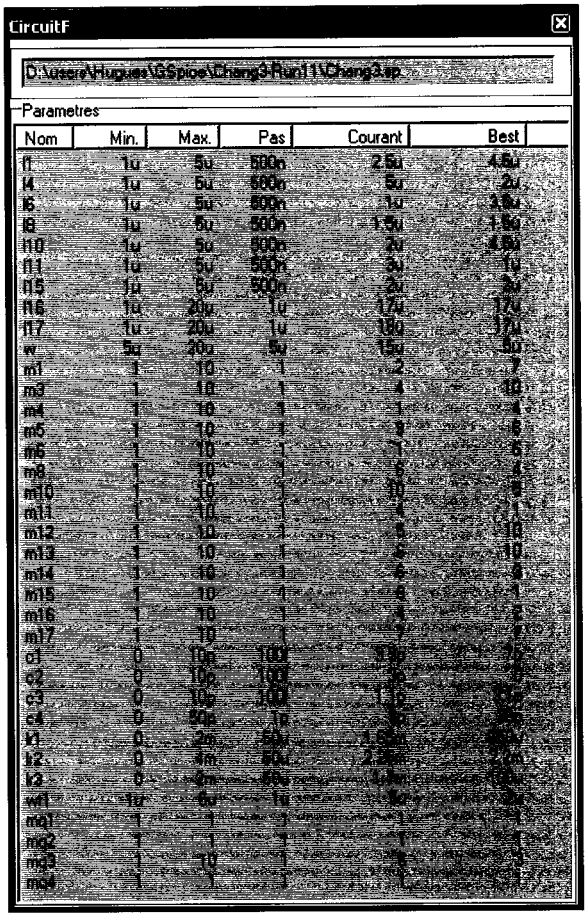


Figure 2.10 Console d'affichage des paramètres optimisables

2.4.3.3 Console des objectifs et de construction du pointage global

La console des objectifs (Figure 2.11) permet de sélectionner, dans la première fenêtre, les nœuds contenus dans les analyses (fenêtre **Mesures**). La deuxième fenêtre de cette console présente les différentes *fonctions de pointage* réalisées pour dicter les objectifs de l'optimisation (fenêtre **Pointeur**). Juste en dessous, on trouve la description de l'*équation d'utilité* qui unit les différents pointages (**Équation**). Enfin, dans la section

complètement en bas de cette console, on trouve les valeurs de pointage globales calculées pour le candidat *courant* ainsi que le *meilleur* rencontré jusqu'à présent.

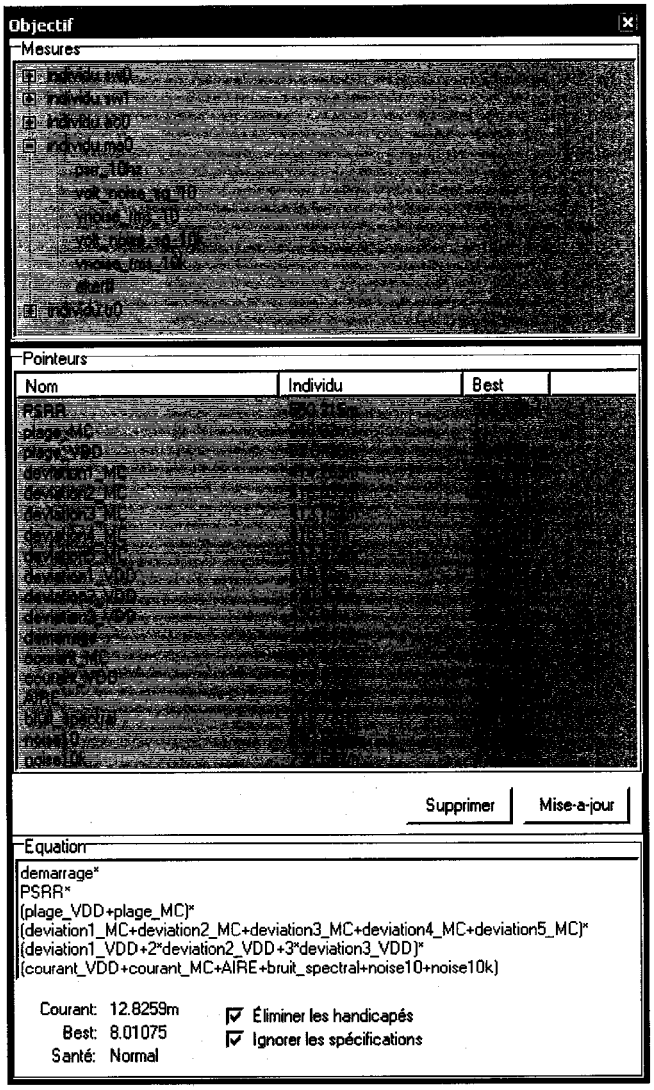


Figure 2.11 Console des objectifs

Lorsqu'un nœud du circuit est doublement cliqué (dans la fenêtre des **Mesures**), il apparaît alors la console de construction de la *fonction de pointage* (module de pointage) comme présenté à la Figure 2.12. Il est a noter que la section 2.4.4 est entièrement dédiée à la description détaillée des multiples fonctionnalités incluses dans cette console.

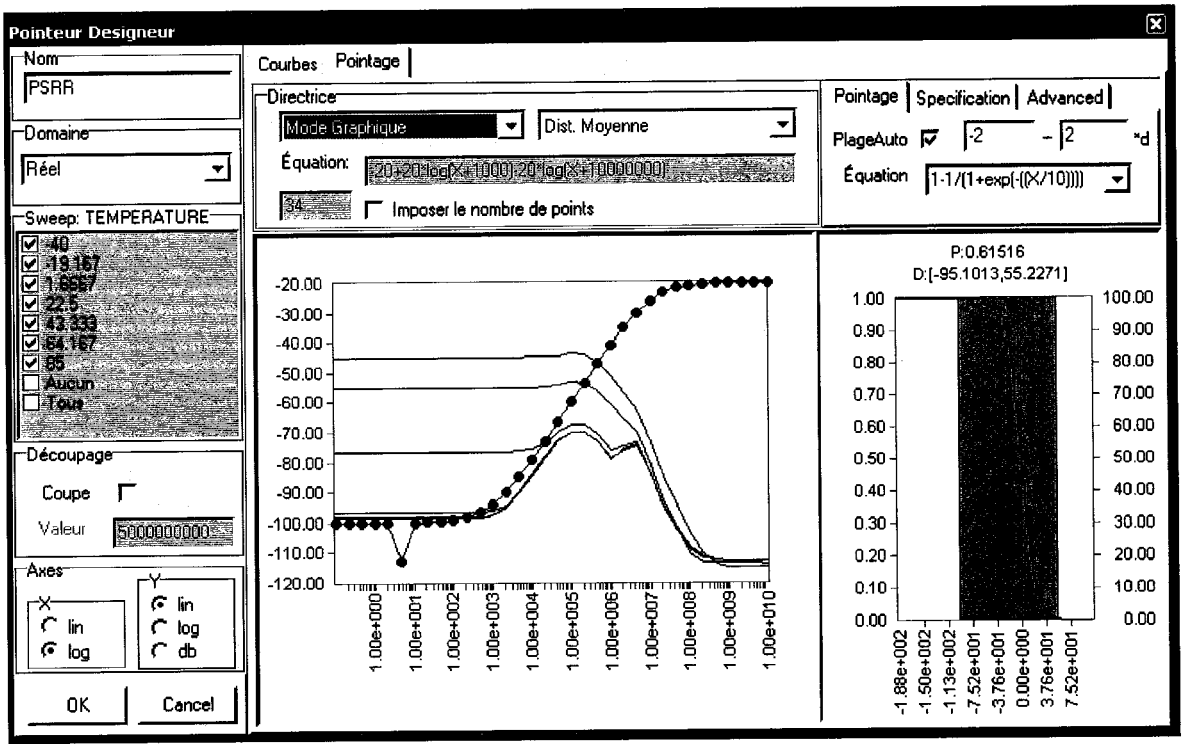


Figure 2.12 Console de construction de la fonction de pointage (module de pointage)

2.4.3.4 Console de configuration du simulateur

La console de configuration du simulateur possède 2 entrées critiques (Figure 2.13). Il est possible d'y spécifier la localisation et donc la version du simulateur qui sera utilisée pour effectuer l'évaluation des circuits en cours d'optimisation. Ensuite, il est possible de spécifier le temps limite alloué à la complétion d'une simulation. Au-delà de cette durée, le processus de simulation d'un circuit (supposément problématique) sera avorté intentionnellement. Enfin, une fenêtre d'affichage (**Statut**) permet de connaître l'historique de l'état d'avancement des différentes simulations exécutées.

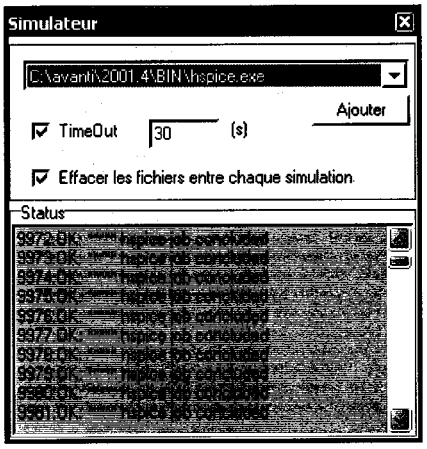


Figure 2.13 Console de configuration du simulateur

2.4.3.5 Console des statistiques du processus d'optimisation

La console d'affichage des statistiques permet de connaître l'historique de la progression du pointage de l'optimisation, afin d'observer sa convergence (Figure 2.14). Ces statistiques présentent les valeurs de pointage du meilleur (courbe supérieure), de la moyenne (courbe médiane) et du plus faible (courbe inférieure) des circuits candidats en fonction des générations (en abscisse).

Il est aussi possible en sélectionnant les différents onglets de la fenêtre de statistiques de connaître l'état de la diversité (**Div.**), de l'écart type (**Dev.**), de la variance (**Var.**) et de la somme cumulée de tous les pointages des individus de la population (**Sum.**). Ces statistiques peuvent être obtenues soit pour la population complète (**Population**) (console de gauche Figure 2.14) ou pour un sous-groupe qui représente une élite (**Best Of All**). On trouve au haut de cette fenêtre le nombre d'individus simulés et le décompte des générations parcourues. Enfin, une boîte à crochet (**log**) permet d'afficher les statistiques selon une échelle linéaire ou logarithmique.

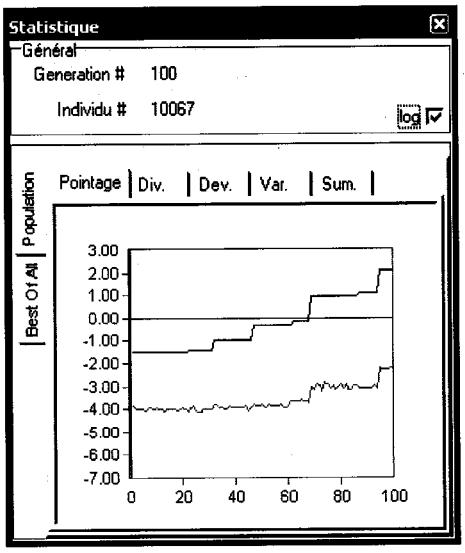


Figure 2.14 Console présentant les statistiques associées au processus d'optimisation

2.4.4 Détails sur le fonctionnement du module de pointage

Le *module de pointage* est un élément clé dans la description des objectifs visés pour une optimisation. Le rôle des *modules de pointage* est de permettre à l'utilisateur du logiciel de dicter deux paramètres importants sur lesquels se basera l'optimisation.

Premièrement, ces interfaces permettent de correctement positionner le but fixé comme objectif de performance (aussi appelé le *vecteur but*[25]). Il est attendu de la part du décideur que celui-ci fixe des objectifs raisonnables pour chacune des performances sous observation. C'est hélas un étape inévitable étant donné que la philosophie adoptée en est une de *programmation par but*.

[25] Les termes vecteur de but, vecteur de référence et droite directrice sont utilisées sans distinction, car ils référent au même concept lié à la programmation par but.

Deuxièmement, il faut déterminer la *fonction de pointage*. Le rôle de la fonction de pointage est de convertir une distance, séparant la mesure du but désiré, en une valeur numérique représentant la *qualité* d'une solution. Le nom *fonction de pointage* signifie essentiellement que cette conversion élimine les problèmes normalement associé à la commensurabilité des performances entre-elles. Cette fonction de pointage dans AGO est de type équation libre. Elle permet ainsi au décideur de dicter n'importe quel type de fonction non-linéaire qui traduit une distance en une valeur de pointage.

La Figure 2.15 présente le flot de traitement sur la mesure de performance des circuits candidats. Nous discutons ensuite de la fonctionnalité de chacunes des étapes présentés à cette figure.

2.4.4.1 Vecteur de référence

Le *vecteur de but* est un élément décisif dans la bonne conduite d'une optimisation. Il doit être choisi de manière judicieuse en intégrant parfois quelques connaissances *a priori* sur le domaine du circuit concerné. Le *vecteur de référence* indique à l'optimisation le but à atteindre, tandis que la *fonction de pointage* indique plutôt le poids qui doit lui être associée.

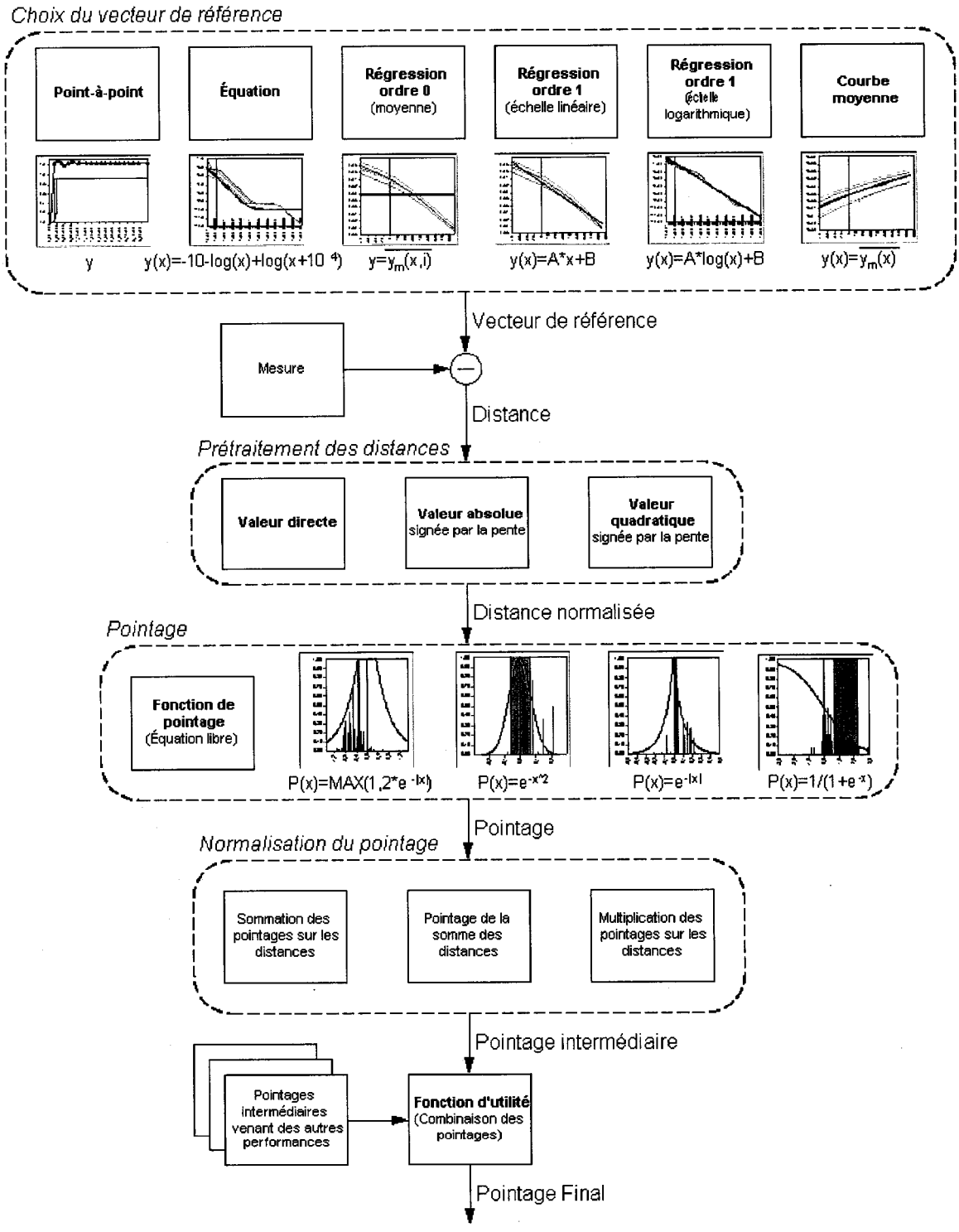


Figure 2.15 Diagramme de flot pour le calcul du pointage dans les modules de pointage

Le vecteur but peut être dicté de 6 manières différentes :

1. **Point-par-point** : le déplacement du vecteur but se fait de manière interactive par des modifications apportées à la main sur la courbe directrice;
2. **Équation** : le vecteur but est décrit par une équation mathématique libre imposée par l'utilisateur.
3. **Régression ordre 0** : correspond à un calcul automatique sur toutes les courbes de mesures. Effectue le calcul de la *valeur moyenne globale* et l'affecte au vecteur but;
4. **Régression ordre 1 (linéaire)** : effectue le calcul de régression de premier ordre sur les courbes de mesures. Les résultats de la régression sont ensuite imposés comme vecteur but;
5. **Régression ordre 1 (logarithmique)** : réalise le calcul de régression de premier ordre sur les courbes de mesures et selon une abscisse logarithmique. Les résultats de la régression sont ensuite imposés comme vecteur but;
6. **Courbe moyenne** : est définie comme la courbe contituée des valeurs moyennes des courbes de mesures pour chaque valeur d'abscisse. Le vecteur but correspond ainsi à la courbe moyenne sur les courbes de mesures.

Bien que toutes ces définitions soient intégrées, les vecteurs buts de type 5 ou 6 sont moins fréquemment utilisés.

2.4.4.2 Normalisation des distances

La normalisation des distances permet à l'utilisateur d'effectuer une première transformation sur les distances mesurées entre le vecteur but et la mesure selon 3 types de transformations qui sont mises à sa disposition:

1. **Valeur directe** : la valeur de la distance est transmise sans modification à la fonction de pointage. Les distances sont soit positives ou négatives selon que les courbes de mesures sont au dessus ou en dessous du vecteur but.
2. **Valeur absolue** : la valeur de la distance est mise en valeur absolue (toujours positive). Le signe de la distance résultante est déterminé par le signe de la pente de la régression de premier ordre sur les courbes de mesure.
3. **Valeur quadratique** : une métrique euclidienne est appliquée sur les distances. Le signe de cette distance est déterminé de la même manière que pour la *valeur absolue*.

La normalisation des distances trouve son utilité dans le fait que sans celle-ci, il est impossible de procéder à certaines optimisations très spécifiques. Par exemple, l'optimisation, en température, de la pente de la tension de sortie d'une référence de tension nécessite absolument la mise en *valeur absolue* des distances. Ceci découle de la nécessité d'un cumul additif des distances afin de promouvoir une dérive en température la plus plate possible. Cependant, dans la majorité des autres cas rencontrés, le décideur opte généralement pour la transmission en valeur directe.

2.4.4.3 Fonction de pointage

Il est normalement conseillé de définir des fonctions de pointage normalisées [0,1]. En effet, cela facilite leur intégration dans l'*équation d'utilité* (section 2.4.4.5) et rend plus simple leurs interactions avec les autres pointages. Surtout lorsque des multiplications entrent en jeux.

Tel qu'il a été présenté à la Figure 2.15, quelques exemples de fonction de pointage sont rapportés. On y distingue quatre des fonctions de pointage les plus couramment utilisées avec ce logiciel (fonction tronquée, fonction gaussienne, fonction exponentielle miroir et fonction sigmoïdale). Il est important de rappeler que l'équation décrivant la fonction de pointage est toujours de type libre. Ainsi, il est possible de recourir à n'importe quelle formule réalisant une conversion *distance-pointage* qui semble judicieuse à l'utilisateur.

Le rôle de la fonction de pointage est de traduire l'écart entre la performance mesurée et le but fixé par l'intermédiaire d'un *indice de performance*. Plus la performance s'approche de son but et plus les pointages associés à ces améliorations devront être élevés.

Bien sûr, il est généralement conseillé d'utiliser des fonctions de pointage normalisées, bien que ce ne soit pas ici une obligatoire formelle.

2.4.4.4 Fonction de normalisation

Étant donné que les mesures de distance se font point par point, l'agglomération des différentes fonctions de pointage, pour une même mesure de performance, reste limitée à un choix restreint d'alternatives. Les différentes valeurs de pointage peuvent être combinées selon l'une des trois méthodes suivantes :

1. **Sommation des pointages sur les distances** : conversion des distances point par point dans la fonction de pointage et *sommation* de toutes les valeurs de pointage obtenues pour toutes les courbes (valide aussi dans les cas d'une courbe multiple);
2. **Pointage sur la somme des distances** : *sommation* de toutes les distances associées à une courbe et conversion de la distance résultante par la fonction de pointage. Lorsqu'il y a présence de courbes multiples, on procède ensuite à la sommation des valeurs de pointage;
3. **Multiplication des pointages sur les distances** : conversion des distances point par point par la fonction de pointage et *multiplication* des valeurs de pointage obtenues pour toutes les courbes.

Bien sûr, la méthode la plus couramment utilisée reste la **sommation des pointages sur les distances**. Cependant, les deux autres méthodes peuvent s'avérer très utiles dans certains contextes bien précis d'optimisation. Un très bon exemple de leur utilité est probablement celui du rejet rapide de candidats handicapés (au niveau d'une de leurs performances).

Une utilisation concrète de la 3^{ième} méthode, correspond au cas où il faut rejeter rapidement les cas d'instabilité apparaissant au niveau du PSRR d'une référence de tension. Il arrive parfois, au niveau des fréquences intermédiaires, que le circuit produise une amplification du bruit en sortie en provenance de son alimentation. Or, l'impact immédiat se traduit par un PSRR qui peut déborder au-delà de la barrière du 0 dB (voir encerclé à la Figure 2.3b) dans la réponse en fréquence. Une fonction de pointage de type *min-max*, combinée avec une multiplication des pointages et placée judicieusement en facteur multiplicatif dans la fonction d'utilité, conduira au rejet programmé des mauvais circuits candidats. L'observation a été faite que les PSRR dépassant le seuil de 0 dB conduisaient à des comportements instables et potentiellement oscillatoires lorsqu'observé dans le domaine temporel. Dans ce cas précis, la caractéristique appréciée de la multiplication des pointages est de conduire le pointage au rejet du candidat en lui assignant une valeur très faible et en faisant tendre ainsi rapidement le pointage global vers une valeur nulle.

2.4.4.5 Fonction d'utilité

La fonction d'utilité est un concept qui est propre au domaine de la *programmation par but*. Elle consiste à utiliser une équation qui combine les différents pointages obtenus pour les performances observées. On retrouve essentiellement dans cette équation les ordres d'importance associés à chacun des critères qui optimisent le circuit. Dans le cas d'une référence de tension, on mentionne aux sections 3.1 et 3.3 les différentes performances observées et optimisées. Or, il n'est pas conseillé de mettre tous

ces critères sur le même pied d'égalité (ex : par une simple sommation). Des phénomènes d'ombrage sont trop enclins à se produire et leur déverminage peut constituer une perte de temps appréciable. C'est pourquoi, la fonction d'utilité basée sur une somme pondérée systématique n'a pas été privilégiée dans cet outil. (section 1.1.2.1)

Il a été jugé préférable de laisser au concepteur le contrôle sur la direction de l'optimisation en lui permettant de structurer l'équation qui combine les différents objectifs associés à son circuit. L'équation suivante présente un exemple de combinaison judicieuse des performances à optimiser pour une référence de tension :

Fonction d'utilité = BOOT*Plage*Déviation* PSRR*(Courant+Surface+ Bruit) (5)

Comme on peut le constater, le démarrage (reflété par la variable *pseudo-booléenne*[26] BOOT), la stabilité en température (Déviation), la plage de tension de sortie (Plage) et l'immunité à l'alimentation (PSRR), toutes les propriétés qui font du circuit une référence de tension, sont directement reflétées dans cette fonction. Le fait de sortir des mesures de performance de la sommation et de les positionner en facteurs multiplicatifs contraint l'optimisation à d'abord rencontrer ces objectifs, avant de se concentrer sur les autres performances comprises dans la sommation. Dans ce sens, cette démarche suit la logique puisqu'il ne sert à rien d'optimiser une référence de tension qui *n'en est pas encore une* ou qui *ne réussit même pas à démarrer toute seule*?

---

[26] Pseudo-booléenne : réfère à une variable continue dont le domaine est compris entre 0 et 1. Concept fortement inspiré et utilisé en logique floue.

3 Mise en application et résultats

Ce chapitre présente l'optimisation d'une référence de tension basée sur la topologie de Chang-Hyeon [58][59]. Cette topologie sera référée dans la présente étude par le nom de *Chang*. Ce circuit de référence de tension a été proposé dans ces articles comme un élément auxiliaire à un PLL. Nous supposons que c'est la raison pour laquelle aucune dimension de transistor n'est relatée, faute de place. Par conséquent, cette référence de tension constitue un excellent banc d'essai pour effectuer une optimisation paramétrique sur les tailles des transistors. Les multiples raisons associées à ce choix seront expliquées à la section 3.1.

3.1 Schéma bloc et schéma électronique

La référence de tension *Chang* constitue vraiment un circuit de choix pour effectuer une optimisation multi-variables, multi-objectifs; voici quelques arguments qui plaident en sa faveur.

Le premier critère qui rend la référence de tension Bandgap si intéressante à optimiser est le fait qu'elle utilise une bonne diversité de composants disponibles en microélectronique, soit : des résistances, des condensateurs, des transistors MOSFET et des diodes (souvent réalisés à partir de transistors bipolaires verticaux parasites).

En second lieu, les multiples boucles de rétroaction présentes au sein de l'architecture du *Chang* en font un circuit très difficile à synthétiser sans une *approche globale*. En effet, les différents modules qui composent le *Chang* ont leurs points de

fonctionnement finement inter-reliés avec les points d'opérations des autres modules auxquels ils sont raccordés. Cette imbrication serrée a normalement de quoi décourager, en première approche, même les bons concepteurs.

En troisième lieu, bien que les références de tension soient des circuits qui semblent simples, la couverture complète de toutes leurs performances exige l'utilisation de l'ensemble des domaines d'analyse offerts par les simulateurs de circuit. Voici certaines des performances à vérifier et à optimiser :

- Stabilité en température de la sortie (TEMPCO)
- Immunité aux variations de l'alimentation (PSRR)
- Capacité à démarrer rapidement et de manière autonome (BOOT)
- Bruit intrinsèque généré à la sortie (NOISE)
- Surface de silicium utilisée (AREA)
- Consommation en puissance (COURANT)

- Plage d'opération de l'alimentation (_VDD)
- Impact des variations sur la température (_TEMP)
- Impact des variations des procédés de fabrication (_MC)

La Figure 3.1 présente le schéma bloc de la référence de tension utilisée. Cette référence de tension est essentiellement constituée de 4 blocs principaux, dont : un filtre passe-bas, un régulateur d'alimentation, une référence de tension basée sur le principe d'extraction de la tension de *Bandgap* et un module qui assiste le démarrage du circuit.

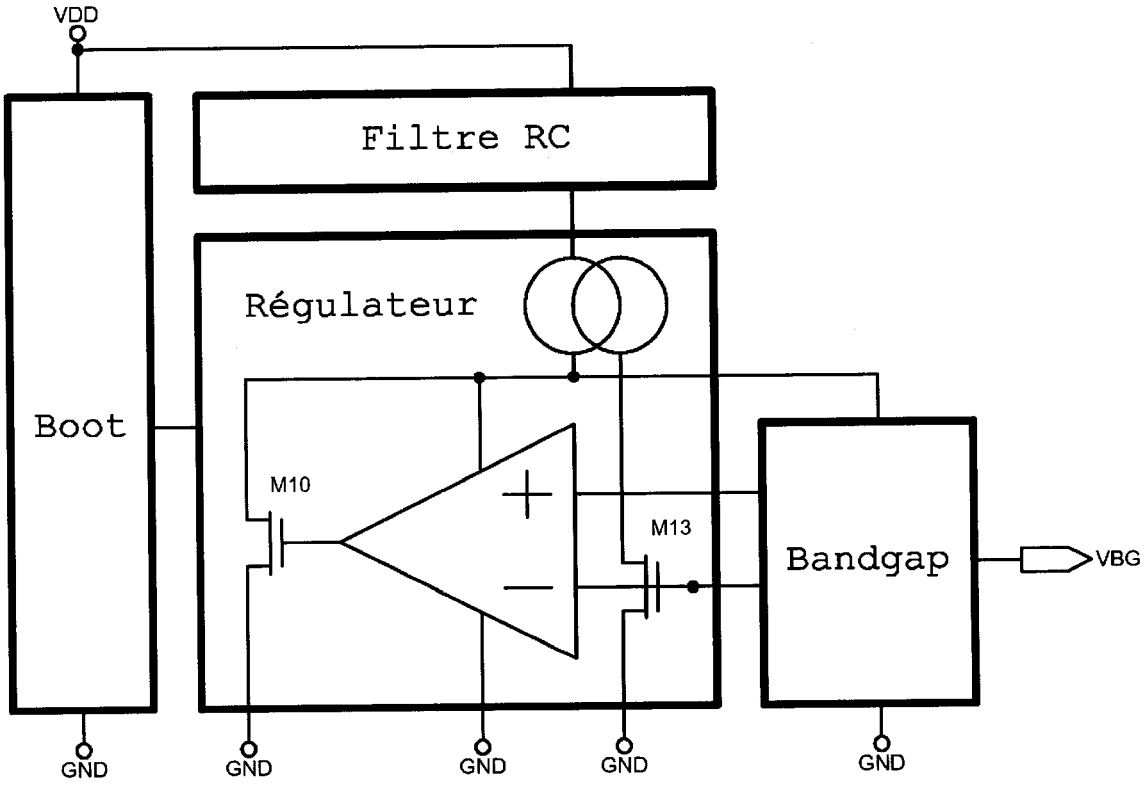


Figure 3.1 Schéma bloc global de la référence de tension *Chang*

Nous allons procéder à un découpage des modules et nous allons expliquer leur rôle au sein de l'architecture de la référence de tension.

La Figure 3.2 présente le schéma électronique simplifié correspondant au schéma bloc de la Figure 3.1. Chacun des modules découpés joue un rôle précis tel qu'indiqué sur le schéma bloc. Le filtre passe-bas est réalisé par un circuit RC simple. Le module de démarrage peut être considéré comme un diviseur résistif qui applique une tension *médiane* afin d'activer ou bloquer un transistor de court-circuit temporaire (M15). Le régulateur est quant à lui subdivisé en deux blocs : un régulateur de courant et un régulateur d'alimentation. Ces deux régulateurs travaillent conjointement afin d'appliquer

une tension régulée au module générateur de *Bandgap*. Enfin, le générateur de *Bandgap* a pour rôle de produire la fameuse tension stable et indépendante aux variations de la température. Tous ces modules travaillent généralement de concert et l'ajustement d'un de ceux-ci engendre nécessairement des impacts sur le fonctionnement des autres. En ce sens, le degré d'interaction des modules entre-eux est élevé.

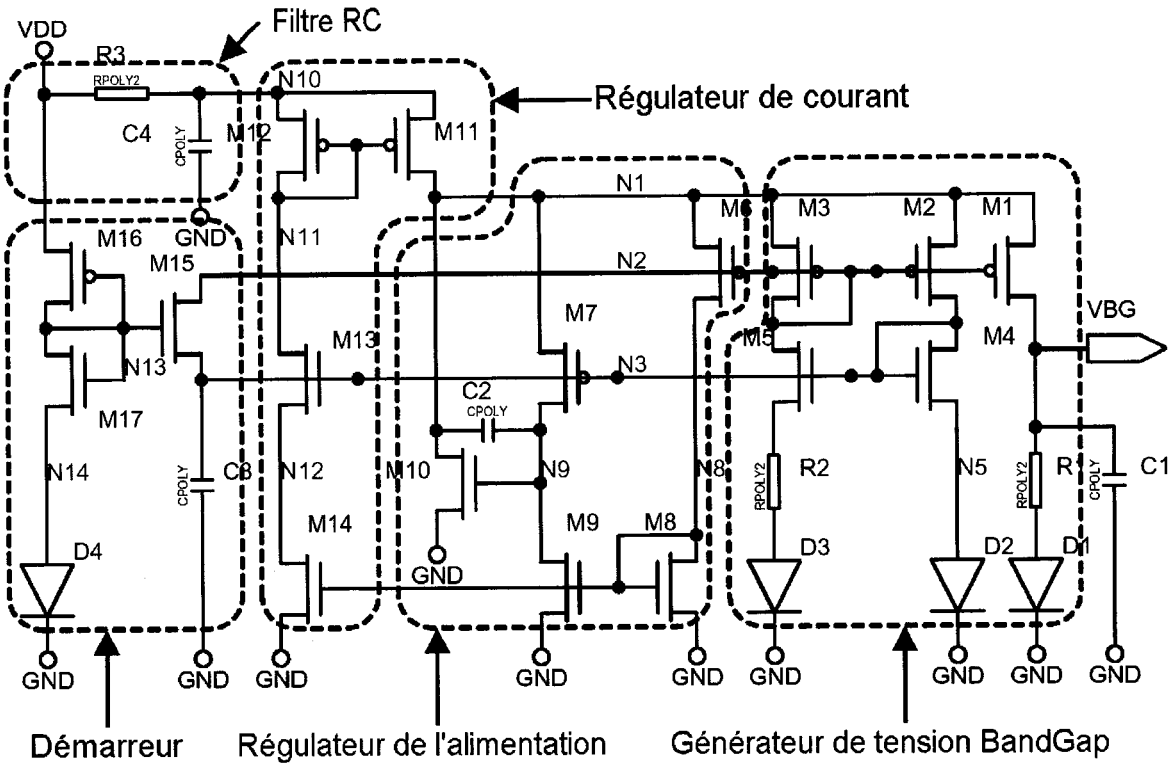


Figure 3.2 Schéma électronique simplifié présentant les différents modules spécialisés de la référence de tension *Chang*

Les sections suivantes présentent plusieurs détails quant aux particularités et au fonctionnement des modules constituant la référence de tension *Chang*.

3.1.1 Générateur de tension Bandgap

Le coeur *Bandgap* (Figure 3.3) est la portion du circuit qui s'occupe de réaliser la tension stable en température. Pour y parvenir, on utilise deux diodes (D2, D3) dans lesquelles circulent des densités de courant différentes. La chute de potentiel *différentielle* entre les deux bornes (N5, N6) tendra à suivre une *progression proportionnelle à la température absolue* (PTAT) [41]. Le miroir de courant formé par M2 et M3 rapportera ainsi un courant PTAT à la branche de sortie, de manière à compenser, à travers R1, la dérive de –2 mV/°C associée à la diode D3.

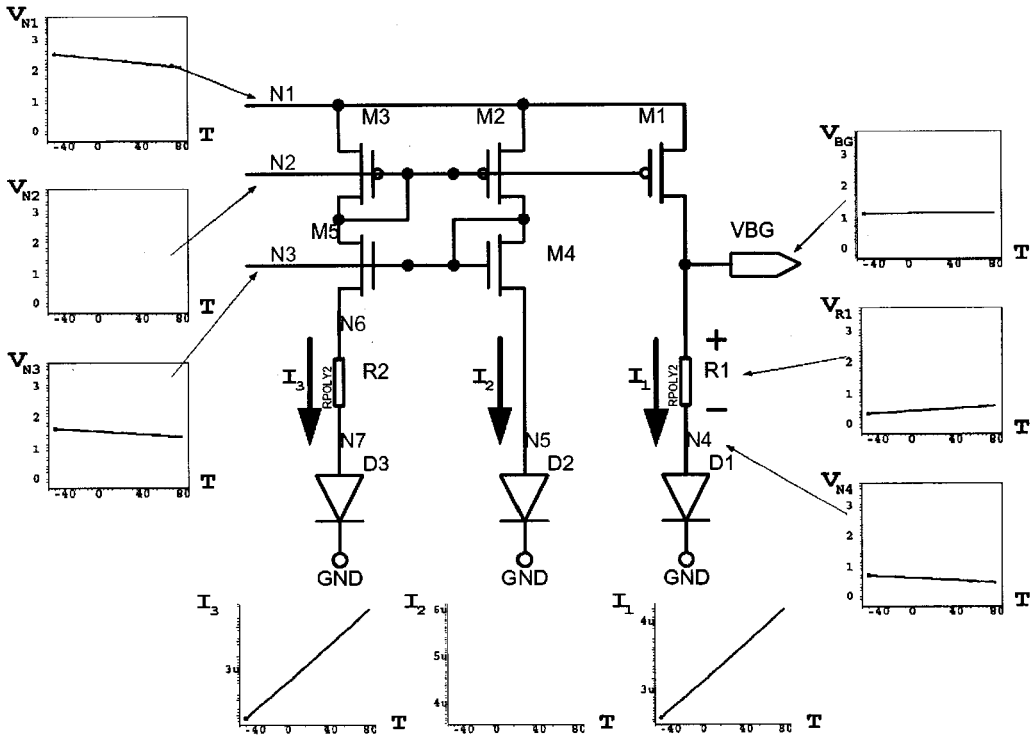


Figure 3.3 Fonctionnement du cœur de la référence de tension *Bandgap*

Étant donné que les résistances R1 et R2 sont volontairement appariées et construites à partir des mêmes matériaux, elles partagent normalement un *coefficient en*

température (TC) presque identique, ainsi qu'une même résistivité. Cet appariement permet d'éliminer, dans les équations analytiques qui décrivent la tension de sortie produite par cette structure, l'apport de certaines non-linéarités accompagnant le comportement en température des résistances.

Le cœur de la référence, dans la configuration suivante, est très sensible aux variations sur son alimentation (nœud N1). Normalement, cette architecture possède une piètre *immunité aux variations de l'alimentation* (PSRR), de l'ordre de –10 dB. Afin d'augmenter cette immunité aux variations pouvant survenir sur l'alimentation (VDD), on fait appel à un circuit de régulation que l'on peut identifier à la Figure 3.2 et qui sera décrit à la section suivante.

3.1.2 Régulation de l'alimentation

Le module de régulation de l'alimentation est essentiellement constitué d'une paire différentielle (M6, M7) qui détecte toutes divergences entre les tensions des deux noeuds N2 et N3. De par la nature du *miroir de Wilson* (M2, M3, M4 et M5) [41], les tensions sur N2 et N3 doivent être essentiellement identiques, une fois le circuit enclenché. Dans le cas contraire, une correction est apportée à la tension de N1 à travers M10. De cette manière, toutes variations faites sur VDD qui déplacerait le miroir de courant Wilson de son point d'opération sera corrigé automatiquement par la boucle de rétroaction formée par le pseudo-amplificateur opérationnel (M6, M7, M8, M9 et M10).

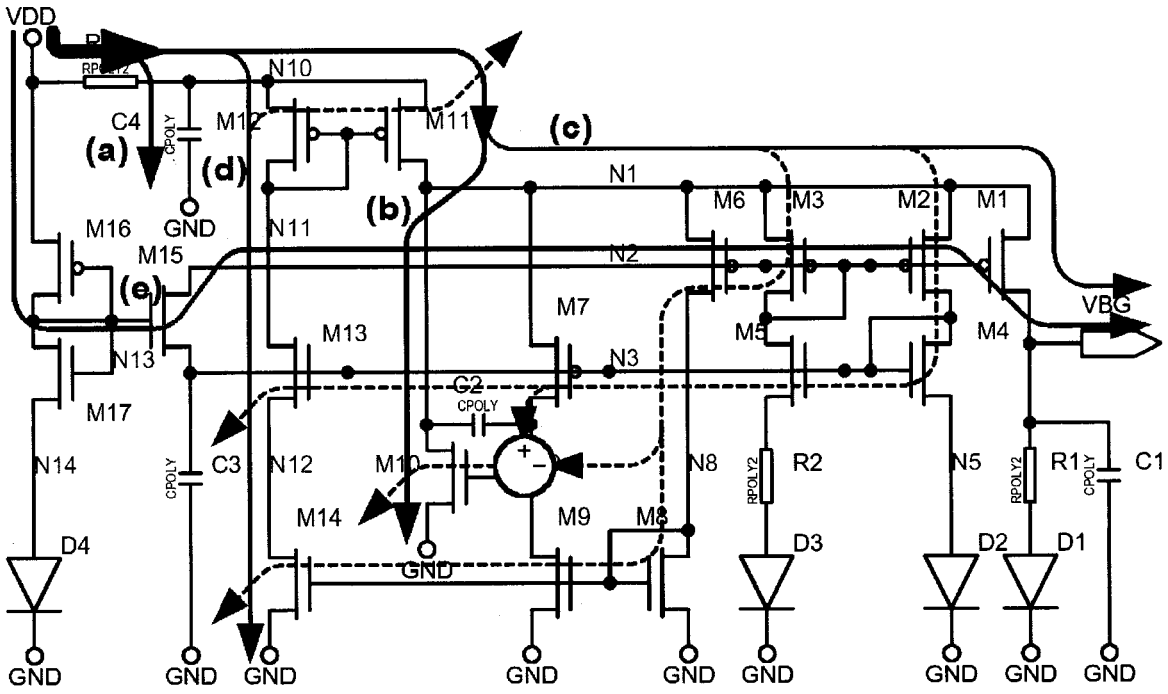


Figure 3.4 Diagramme qui illustre le fonctionnement de la régulation de l'alimentation

La Figure 3.4 illustre, le fonctionnement du régulateur de l'alimentation. Un bruit provenant de l'alimentation doit premièrement passer par un filtre RC, qui agit en filtre passe-bas et élimine ainsi la composante à haute fréquence du bruit; trajet (a). Ensuite, lorsque cette fluctuation de l'alimentation parvient à atteindre le nœud N1, le miroir de Wilson perçoit le changement, puisque ses deux tensions (N2 et N3) se trouvent perturbées. L'information se propage ainsi jusqu'au transistor M10 qui réalise alors la déviation du courant en supplément; trajet (b). Cette action a pour effet d'abaisser la tension au nœud N1 et de remettre le générateur de tension *Bandgap* à son point d'opération optimal. Il est certain qu'un résidu de bruit peut quand même parvenir jusqu'à la sortie, mais il est, dans cette condition, grandement atténué; trajet (c).

Les transistors M13 et M14 jouent aussi un rôle dans la régulation de l'alimentation. En effet, ces deux transistors ont pour rôle de contrôler le courant entrant dans la référence de tension. Si celui-ci est trop élevé, une rétro-action se produit aussitôt afin de corriger l'apport en courant; trajet (d).

Malgré le circuit de régulation, il subsiste un chemin indésirable par lequel les variations de l'alimentation peuvent se propager jusqu'au cœur de la référence de tension. Il s'agit du passage par la portion du circuit qui s'occupe du démarrage de la référence de tension; trajet (e). Le circuit de démarrage fait l'objet de la prochaine section.

3.1.3 Circuit de démarrage

Les références de tension possèdent normalement 2 points d'opération stables. Le premier est celui où *aucun* courant ne circule dans le circuit et la tension de sortie vaut zéro. Le second point d'opération, celui désiré, correspond à une tension de référence en sortie de 1.23V. Pour forcer le démarrage de cette référence de tension, un diviseur de tension a été ajouté (M16, M17 et D4) de manière à court-circuiter les nœuds N2 et N3 jusqu'à ce que leurs différences de potentiel avec N13 deviennent inférieures à la tension de seuil (V_{TH}) du transistor M15. Lorsque cet état est atteint, le transistor M15 devient ouvert et donc désactivé.

La connexion du transistor M15 à deux nœuds critiques de la référence de tension ouvre cependant une voie aux perturbations venant de l'alimentation. Les fluctuations de l'alimentation peuvent, à travers la capacité parasite grille-drain/source de M15, se frayer

un chemin et venir perturber le PSR de la sortie; trajet (e) (Figure 3.4). Chemin par lequel la régulation ne peut ensuite que maladroitement effectuer une compensation efficace. Mais heureusement, le transistor M16 possède normalement une impédance élevée, ce qui réduit de beaucoup l'ampleur de la brèche. Cette valeur d'impédance élevée est d'autant plus encouragée que le circuit est optimisé pour consommer peu de courant.

3.1.4 Contraintes structurelles

La Figure 3.5 présente les contraintes structurelles imposées (appariements entre les transistors) afin de réduire la dimensionnalité de l'espace de recherche tout en rendant plus significatif les circuits qui vont être investigués.

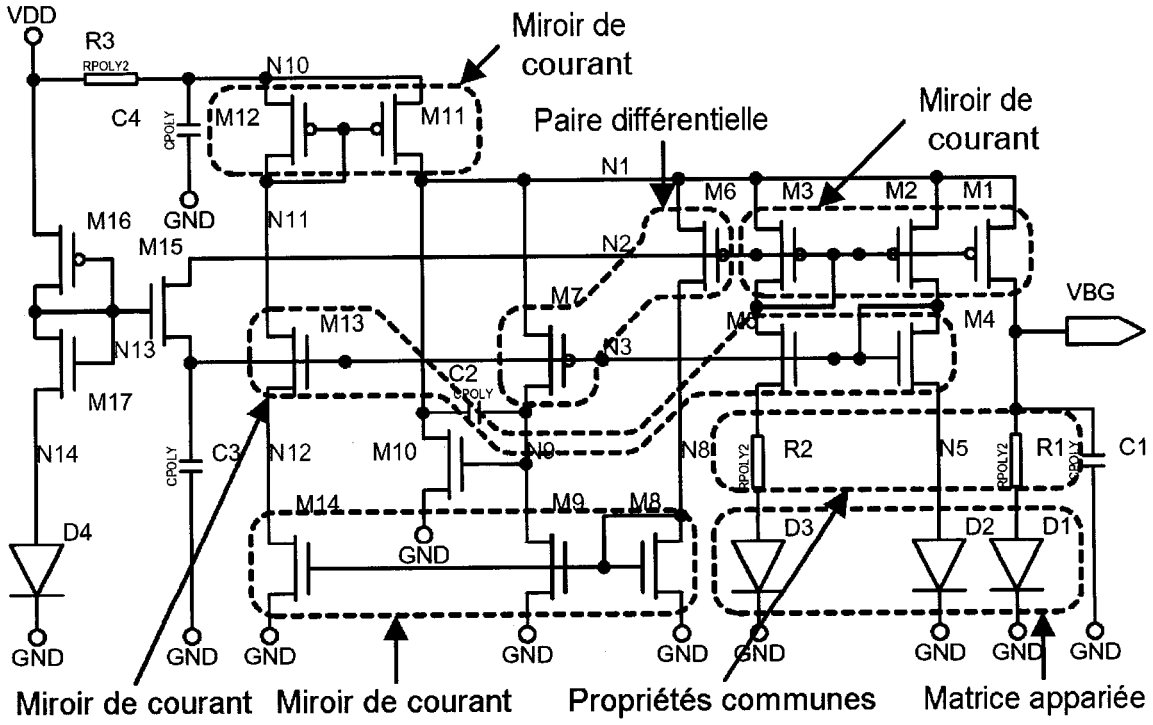


Figure 3.5 Contraintes structurelles imposées aux composantes de la référence de tension

Les cas classiques d'appariement concernent les *miroirs de courant* et les *paires différentielles*. Ces appariements prennent effet, dans les transistors MOS, sous la forme de longueurs et de largeurs de dimensions partagées. Dans les références de tension *Bandgap*, il est aussi fortement conseillé d'adopter ces stratégies pour l'appariement des diodes et des résistances. Ces considérations ont pour effet de réduire significativement les variations relatives venant des procédés de fabrication et pouvant affecter les propriétés électriques des composants. Cependant, dans le cadre exploratoire de ce mémoire, le cas du miroirs de courant Wilson s'est vu uniquement imposé par un appariement sur la longueur, laissant optimisable le facteur multiplicatif des différents transistors le constituant.

De plus, l'appariement offre un net bénéfice lorsque vient le temps de procéder à la réalisation des dessins des masques. En effet, des structures régulières qui se juxtaposent facilement en damier permettent l'application d'une méthode de placement plus simple et plus efficace.

3.2 Transcription en *netlist*

Le listage complet de la référence de tension *Chang* est fournit en Annexe I sous le nom de "Chang.sp". Une note importante à rapporter à propos de cette description textuelle du circuit concerne le retranchement des analyses qui sont incluses dans un fichier à part et portant le nom de "Chang.lib". Cette stratégie reflète les explications de la section 2.3.4.

Les paramètres optimisables sont identifiés et capturés dans la *netlist* par la déclaration suivante :

```
.PARAM param# = opt#(nom,min,max,pas)
```

Listage 3.1 Déclaration d'un paramètre optimisable.

Les *paramètres optimisables*, à l'instar des définitions standards de paramètres, sont détectés par le marqueur `opt`, où le # est un identificateur de groupe qui peut être un champ vide ou une valeur alpha-numérique quelconque. Le champ *nom* est affecté comme la valeur nominale du paramètre. Les champs *min* et *max* sont les bornes inférieures et supérieures allouées à l'optimisation. Enfin, le champ *pas* est le pas d'exploration sur la plage spécifiée. La nomenclature `opt` est compatible à 100% avec la définition mise en place dans HSpice pour effectuer des optimisations utilisant leur propre algorithme interne (commande `OPTIMIZE`). Bien que la section 1.2.2.1 présente le type d'algorithme utilisé par ce simulateur, nous invitons le lecteur à consulter le chapitre 13 du manuel de HSpice [108] pour plus de détails sur l'utilisation de celui-ci.

Le seul défaut que l'on puisse reprocher à cette définition de paramètre optimisables concerne la limitation à des incréments linéaires sur les plages. Heureusement, il est possible d'effectuer des opérations mathématiques complexes sur ces paramètres après coup et ainsi se libérer de cette contrainte.

```
.PARAM param_exposant = opt(1,0,5,1)
R1 N1 GND R='pwr(10, param_exposant)'
```

Listage 3.2 Transformation mathématique sur un paramètre optimisable.

Le Listage 3.2 présente un exemple de comment optimiser une résistance entre 1Ω (10^0) et 10kΩ (10^5) à raison d'une décade par incrément.

Une gamme d'autres opérateurs mathématiques sont disponibles à l'intérieur même de HSpice et nous référons le lecteur au chapitre 7 du manuel de HSpice [108] pour de plus amples informations.

3.3 Formulation de l'optimisation

La première chose importante à faire avant de se lancer dans une optimisation, c'est d'identifier et de répertorier les analyses et les mesures nécessaires afin de correctement cerner le circuit que l'on désire optimiser. Si ce travail est mal fait ou qu'il y a omission de certaines analyses vitales, l'éventualité de l'apparition de circuit *monstreux* est grandement encouragée (section 4.2). Trois domaines sont disponibles pour caractériser un CA : le domaine paramétrique (V, T°, I, etc.), le domaine fréquentiel (Hz) et le domaine temporel (secondes). Normalement, un concepteur alterne entre ces différents domaines au cours de la construction de son CA. Dans notre cas, il est nécessaire de bien définir toutes les analyses et les mesures qui doivent caractériser le circuit à optimiser. Ce sont ces mesures qui seront *simultanément* observées et évaluées au cours de l'optimisation par le logiciel. Toute erreur à ce niveau pourrait faire diverger l'optimisation du circuit recherché. Au même titre, un concepteur qui base ses modifications sur le circuit à partir d'analyses de performance trompeuses réalisera fort probablement un circuit non-fonctionnel dans la réalité. Cependant, contrairement au concepteur, la correction de l'analyse erronée en cours d'optimisation ne peut se faire ici

sans certains désagrément, dont celui de devoir généralement redémarrer l'optimisation depuis le départ.

Comme on peut le constater dans la Figure 3.6, les différentes analyses employées produisent toutes les mesures de performances énoncés au début de ce chapitre.

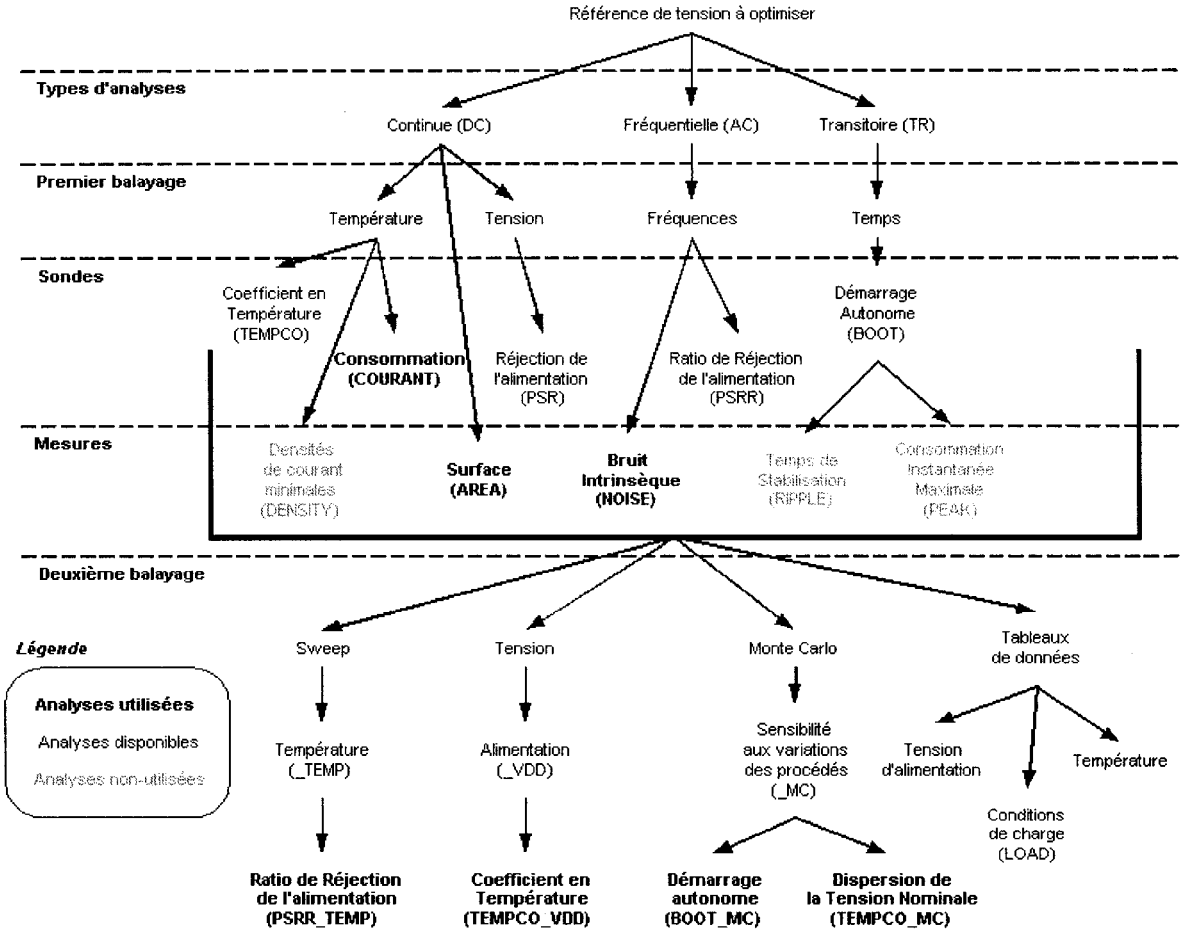


Figure 3.6 Structure hiérarchique des analyses et des mesures nécessaires à l'optimisation d'une référence de tension typique

Cependant, on y discerne, en plus, l'analyse de second balayage qui a pour rôle de valider une performance selon une variable de balayage indépendante supplémentaire, tel

que : la température, l'alimentation, les variations des procédés ou une combinaison quelconque de ceux-ci à travers un tableau de données. L'ajout de cette dimension additionnelle est importante pour caractériser par exemple l'immunité à l'alimentation (PSRR) pour différentes conditions d'opération en température (_TEMP).

Un fois les analyses bien structurées et les seconds balayages les plus importants adoptés, il est possible de procéder aux prochaines étapes nécessaires à la définition de l'optimisation du CA.

3.3.1 Les balises technologiques

Les balises technologiques sont les valeurs limites permises ou allouées pour chaque composant. Par exemple, pour le transistor MOS, il faut définir la longueur du canal (L), sa largeur (W) et le facteur multiplicatif de réplication (M), pour la mise en parallèle.

Bien sûr, il faut savoir que pour les MOS de la technologie TSMC utilisée, le fichier technologique segmente le modèle en 12 secteurs selon la longueur et la largeur de canal des transistors (voir Figure 3.7).

Afin de limiter les erreurs pouvant survenir aux jonctions de ces domaines de modélisation, il est préférable de contraindre certains axes d'exploration. Pour les MOS, le choix sur les largeurs (W) a été fortement discrétisé avec seulement 4 valeurs allouées : 5µm, 10µm, 15µm et 20 µm. Cette décision est en partie motivée par le fait que le facteur de multiplication (M) peut se substituer au rôle de la largeur.

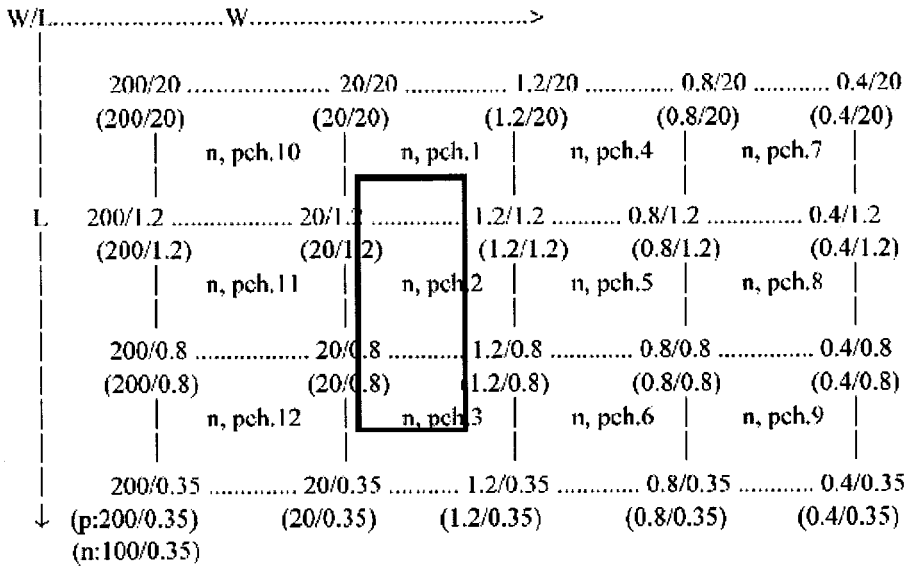


Figure 3.7 Division du modèle des transistors MOS pour la technologie TSMC 0.35μm

Ainsi, normalement l'axe d'exploration de la longueur (L) peut aller jusqu'aux limites technologiques en allant de 0.35 μm jusqu'à 20 μm, à moins d'avis contraire de la part du concepteur. La zone préconisée pour l'optimisation des MOS est présentée à la Figure 3.7 par le *rectangle* qui circonscrit les largeurs (W) dans le domaine situé entre 5μm et 20μm par pas de 5μm et des longueurs (L) dans le domaine situé entre 1μm et 5μm par pas de 0.5μm.

Enfin, il fût observé, avec l'expérience acquise par la conduite de plusieurs optimisations, qu'il est parfois intéressant de contraindre ou de resserrer certaines plages paramétriques pour des composants particuliers du circuit. En effet, cette opération permet à l'optimisation d'éviter d'explorer des pans de l'espace associés à de fortes probabilités de mauvais fonctionnement de la part du circuit. De même, lorsqu'un paramètre optimisé atteint une des bornes (inférieure ou supérieure), il est conseillé au

décideur de corriger la situation en augmentant ou en réduisant la borne concernée à une valeur plus lointaine. Cette action permet ainsi d'offrir une nouvelle marge de manœuvre à l'outil d'optimisation.

Le schéma de la Figure 3.8, présente la configuration pour l'optimisation de la référence de tension. La nomenclature utilisée est la suivante : « [X .. Y] : Z » ; où X représente la borne inférieure du paramètre, Y représente la borne supérieure et Z représente le pas entre ces deux extrêmes. Comme on peut le constater dans le schéma, certaines connaissances *a priori* ont été introduites. Les appariements ont été décidés ainsi afin de réduire judicieusement l'espace des solutions.

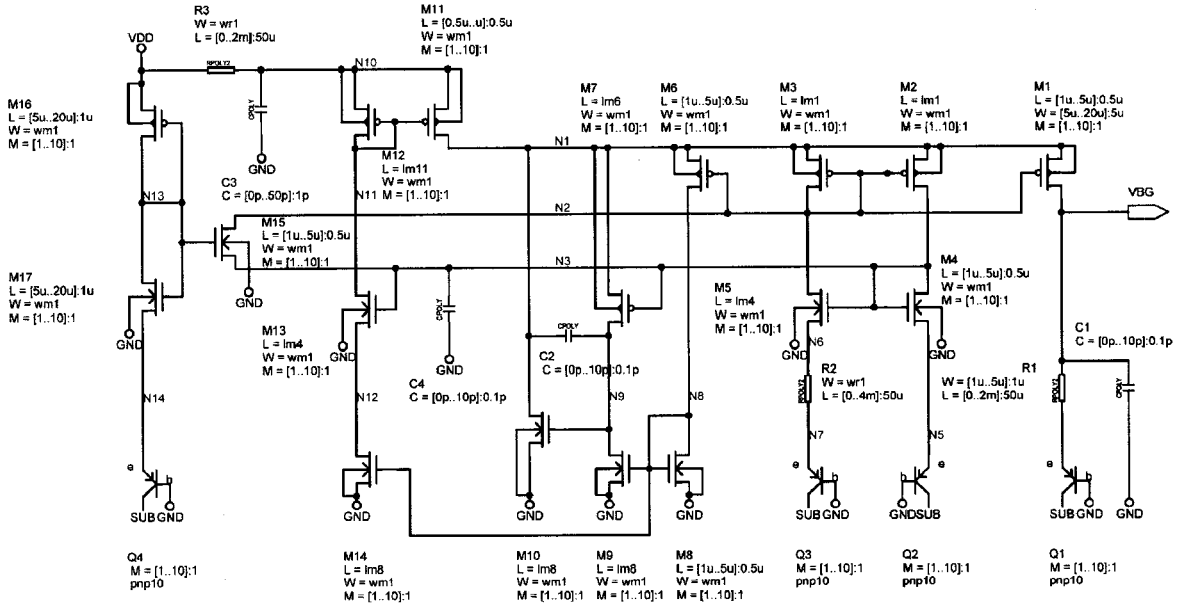


Figure 3.8 Schéma électronique complet du *Chang* incluant les plages paramétriques allouées

3.3.2 Les analyses de performance et les équations de pointage

Cette section présente la description des performances analysées et optimisées par l'outil AGO pour la référence de tension basée sur la topologie de *Chang*. On présente au Tableau 3.1, les plages de caractérisation pour les différents paramètres environnementaux pouvant affecter la référence de tension. Tandis que le Tableau 3.2 présente la nomenclature utilisée afin de correctement identifier les différentes performances analysées et optimisées.

Tableau 3.1 Conditions de caractérisation du circuit

Paramètre	Symbole	Nominal	Plage
Tension d'alimentation nominale	VDD	3.3V	3.0 V @ 3.6 V
Tension de sortie nominale	VBG	1.23 V	-
Température d'opération	T	25 °C,	-40 °C @ 85 °C
Modèle de transistors utilisés	MC	Monte Carlo	5 candidats
Charges capacitives en sortie	CL	0 pF	-
Charges résistives en sortie	RL	$\infty\ \Omega$	-

Tableau 3.2 Nomenclature associée aux performances principales du circuit

Définition des performances	Symbole	Unité
Taille approximative du circuit	AREA	um/côté
Dispersion de VBG sur 30 tirages Monte Carlo @ 25°C	σVBG	mV
Valeur de la tension nominale @ 25°C	VBG	V
Courant consommé par le circuit @ 25°C	COURANT	µA
Stabilité en température de VBG (-40°C à 85°C) @ VDD=(3.0, 3.3, 3.6)[V]	TEMPCO	ppm/°C
Temps de démarrage pour une rampe d'alimentation de 10µsec @ 25°C	BOOT	µs
Réjection des fluctuations sur l'alimentation @ 25°C	PSRR	dB
PSRR @ 100Hz @ 25°C	PSRRBF	dB
PSRR @ 10kHz @ 25°C	PSRRMF	dB
PSRR @ 1MHz @ 25°C	PSRRHF	dB
Bruit intrinsèque intégrée de 0.1Hz à 10Hz @ 25°C	NOIBF	μV_{RMS}
Bruit intrinsèque intégrée de 10Hz à 10kHz @ 25°C	NOIMF	μV_{RMS}

Le Tableau 3.3 présente en détail la configuration des modules de pointage. On y présente entre autre : le vecteur but utilisé, la fonction de pointage associée à ces performances et tous les autres paramètres qui configurent chacun des modules de pointage.

Voici l'équation d'utilité utilisée pour combiner les différents modules de pointage afin de produire le pointage global :

```
BOOT
*(PLAGE_MC+PLAGE_VDD)
*PSRR
*(TEMPCO_MC1+TEMPCO_MC2+TEMPCO_MC3+TEMPCO_MC4+TEMPCO_MC5)
*(TEMPCO_VDD1+TEMPCO_VDD2+TEMPCO_VDD3)
*(COURANT_MC+COURANT_VDD+AREA+NOISE+NOIBF+NOIHF)
```
(6)

Afin de bien estimer l'aire occupée par le circuit, une fois implémenté sur le silicium, il est important de réaliser quelques approximations. De plus amples détails sont fournis à la section AREA de la bibliothèque des analyses présentée au Listage 4.2 de l'Annexe I.

Étant donné que les résistances sont implémentés par un ruban de polysilicium, ce ruban doit être découpé en plusieurs segments de longueur identique afin de ne pas occuper trop d'espace. Un ajout de 2 µm à la largeur assure une séparation sécuritaire entre les rubans, ce qui permet d'éviter tout court-circuitage lors de la fabrication. Une marge de sécurité est aussi appliquée au transistor MOS en multipliant par un facteur 2 la surface occupée par la grille.

Tableau 3.3 Configuration des objectifs spécifiques à chacune des performances mesurées et optimisées

Type de performance	Échelle en ordonnée	Vecteur but (2.4.4.1)	Détail sur le vecteur but {X=abscisse: *Hertz*}	Normalisation de la distance (2.4.4.2)	Fonction de pointage (2.4.4.3) {x=*distance*}	Normalisation du pointage (2.4.4.4)
BOOT	Lin	1	POSITIONNEMENT GRAPHIQUE À 1.23 [V]	1	$e^{-(x/0.5)\ ^2}$	1
PLAGE_MC	Lin	2	1.23 [V]	1	$MIN(1,2*e^{-\|x/0.5\|})$	1
PLAGE_VDD	Lin	2	1.23 [V]	1	$MIN(1,2*e^{-\|x/0.25\|})$	1
TEMPCO_MC[1..5]	Lin	3	-	2	$e^{-\|x/0.025\|}$	1
TEMPCO_VDD[1..3]	Lin	3	-	2	$e^{-\|x/0.025\|}$	1
PSRR	Lin	2	-20+20*log(X+1000) -20*log(X+10000000)	1	$1-1/(1+e^{-x/10})$	1
COURANT_MC	Log	2	**-4.5** = 31.6 [uA]	1	$1-1/(1+e^{-x/0.25})$	1
COURANT_TEMP	Log	2	**-4.5** = 31.6 [uA]	1	$1-1/(1+e^{-x/0.25})$	1
AREA	Log	2	**-3.7** = 200 [um]	1	$1-1/(1+e^{-x/0.5})$	1
NOISE	Log	2	-4-0.5*log(X)	1	$1-1/(1+e^{-x})$	1
NOIBF	Log	2	**-4** = 100 [uV_{RMS}]	1	$1-1/(1+e^{-x/0.5})$	1
NOIHF	Log	2	**-4** = 100 [UV_{RMS}]	1	$1-1/(1+e^{-x/0.5})$	1

Enfin, les transistors bipolaires verticaux parasites avec des émetteurs de 10μmX10μm occupent systématiquement 42μmX42μm quand on tient compte de leurs collecteurs.

Bien sûr, afin d'avoir une meilleure représentation de la surface occupée, on calcule le radical de cette surface afin de connaître la mesure par côté du circuit, comme si celui-ci était dessiné à l'intérieur d'un carré (Tableau 3.4).

Tableau 3.4 Calcul approximatif de l'aire occupée par les composantes du circuit

Type de composantes	Aire directe	Aire reportée
Résistance (en ruban)	L_R*W_R	(L_R'*M_R*(W_R+2μm))
Transistors MOSFET	L_M*W_M*M_M	2*L_M*W_M*M_M
Transistors Bipolaires	10μm*10μm*M_B	42μm*42μm*M_B
Capacité	L_C*W_C	150[fm^2]*((C-2.19[fF])/133.5[fF])

3.4 Configuration de l'algorithme génétique

Les optimisations présentées dans ce mémoire ont été réalisées par un algorithme génétique de type *Steady-State* avec les paramètres de contrôle suivants :

- Taille de la population = 100 individus **(popsize)**
- % de recombinaison = 80% **(pcross)**
- % de mutation = 3% **(pmut)**
- % de recouvrement de population = 90% **(prep)**
- Critère de fin = 100 générations **(ngen)**

De plus amples détails sur les diverses autres variables de contrôle propres à l'algorithme utilisé se trouvent dans [101]. Les valeurs des paramètres de contrôle présentés *ne sont pas ceux fournis par défaut avec l'algorithme*, mais ils ont été adopté au fil du temps à cause de leur bonne convergence pour l'AG. En effet, plusieurs tentatives d'optimisations ont été réalisées avec différentes valeurs de paramètre de contrôle; cette configuration semble converger rapidement et apporter de bons résultats pour la synthèse de circuits. Bien sûr, étant donné la nature onéreuse en temps de calcul de l'optimisation de CA, de plus amples investiguations devraient être réalisés afin de mieux déterminer l'optimalité des valeurs des paramètres de contrôle en ce qui a trait à la convergence de l'algorithme.

Afin de décrire les principales variables énoncées, disons simplement que le taux de recombinaison concerne le pourcentage de la population qui pourra s'accoupler. Le taux de mutation signifie la portion de la population qui sera affectée par une modification aléatoire de leurs gènes. Enfin, le taux de recouvrement de la population est la fraction des individus produits (enfants) qui composera la génération suivante. Ainsi, chaque génération compte toujours le même nombre d'individus (100). Mais sur ce nombre, 10% de cette population (généralement ceux prévilégiés par la fonction de coût) proviendra de la génération précédente.

La simulation des circuits candidats est réalisée, en moyenne, en moins de 6 secondes. Cependant, le délai complet de l'évaluation est réalisé, en moyenne, à l'intérieur de 10 secondes. Ce délai inclut entre autre le temps de chargement des fichiers et le temps de calcul des différents pointages par AGO.

Bien sûr, pour chacun des 10 essais réalisés, le démarrage d'une nouvelle optimisation est réalisé à partir d'un nouveau germe choisi au hasard.

3.5 Compilation des résultats obtenus par optimisation

Dix tentatives d'optimisation ont été réalisées avec exactement la même configuration afin de démontrer la capacité de l'algorithme de converger à l'intérieur d'une contrainte sur le nombre de simulations allouées comme effort de calcul (100 individus * 100 générations ≤ 10000 simulations).

La Figure 3.9 présente une compilation de la convergence des meilleurs candidats au cours de chacun de ces 10 essais et le Tableau 3.5 présente la compilation des résultats pour ces essais. Un détail plus complet de ces statistiques de convergence est présenté à l'Annexe III. De plus, à l'Annexe V, on présente de manière détaillée toutes des courbes de performance rapportées par le Tableau 3.5.

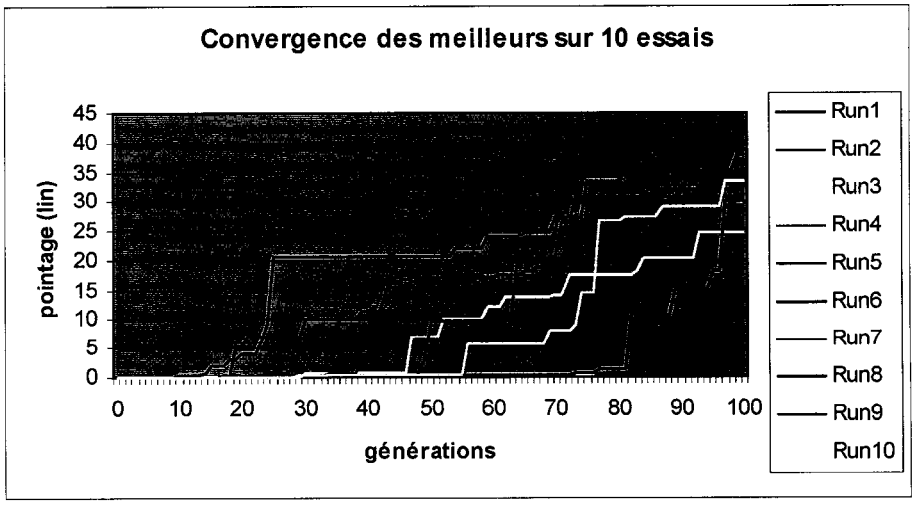


Figure 3.9 Compilation statistique de la convergence des meilleurs candidats sur 10 essais

La Figure 3.9 permet d'observer que l'AG commence sa recherche avec une population essentiellement non-adaptée au problème, puisque les pointages associés à la population sont très faibles. Cependant, au fil des générations et par le processus de sélection naturelle, l'AG converge graduellement par des soubres-sauts typiques aux petites et grandes découvertes faites à la suite de multiples essais et erreurs. De plus, il est possible de voir de manière claire les différents pointages finaux auxquels ont convergé les circuits optimisés.

Le Tableau 3.5 présente le sommaire des performances obtenues pour 10 optimisations réalisées sur le même circuit. Les *valeurs paramétriques* associées à ces circuits sont présentées à l'Annexe IV.

Tableau 3.5 Compilation des résultats de performance pour 10 essais d'optimisations

Essai #	Pointage	VBG (V)	σVBG (mV)	TEMPCO (ppm/°C)	COURANT (µA)	PSRRBF (dB)	NOIMF (V_{RMS})	BOOT (µsec)	AREA (µm)
1	37.82	1.194	58.2	83.1	34.3	-80	337	**<55.5** (1) =3ms	**220**
2	**39.24**	1.209	36.1	**7.28**	30.8	-94	410	<294 (3) >4ms	286
3	24.58	1.274	49.3	83.5	**20.5**	-90	460	<889	258
4	35.08	**1.233**	41.5 (4)	57.1	25.1	**-106**	219	<720	239
5	30.69	1.156 (2)	42.9 (2)	51.9 (2)	44.3	-65	**144**	<419 (4) >3ms	308

Essai #	Pointage	VBG (V)	σVBG (mV)	TEMPCO (ppm/°C)	COURANT (µA)	PSRRBF (dB)	NOIMF (V_{RMS})	BOOT (µsec)	AREA (µm)
6	**38.43**	1.215	39.8	33.5	43.8	-90	295	instable	298
7	26.77	1.323	33.9	25.4	41.2	-95	318	**194.5**	**220**
8	29.99	1.263	33.7	50.0	**17.8**	**-105**	275	1656	316
9	32.13	1.217	46.6	31.5	25.1	-96	284	543	302
10	33.37	**1.226**	**21.1**	35.9	28.9	-93	**149**	640	329

On remarque en caractères **gras,** les différentes performances qui se démarquent par rapport aux autres résultats. Les nombres entre parenthèses identifient le nombre de candidats Monte Carlo, sur 30 essais aléatoires, qui ont échoué le test. De manière simplifiée, l'analyse Monte Carlo est une *analyse statistique* qui permet de connaître la robustesse d'un circuit (section 2.3.5). L'interprétation communément faite pour un tirage de 30 candidats est que s'ils démontrent tous une performance acceptable alors il est probable que plus de 80% des circuits fabriqués auront une probabilité supérieure à 99% de fonctionner correctement. Ainsi, les solutions candidates fautives à ce test sont normalement catégorisés comme possédant un *handicap majeur*, et ce malgré leur bonnes performances sur les autres plans. Pour obtenir plus de détails sur ces différentes défaillances, nous vous référons à l'Annexe V.

4 Discussion

4.1 Discussion à propos des résultats d'optimisation

Un circuit fonctionnel est toujours préférable à un circuit qui ne fonctionne pas. Or, le premier constat à porter au Tableau 3.5 concerne les essais 1, 2, 4, 5 et 6 qui révèlent certaines imperfections au niveau fonctionnel. En effet, les essais 1, 2, 5 et 6 présentent, à travers les analyses exhaustives de la validation (voir Annexe V), certaines difficultés au niveau du démarrage autonome en regard aux variations des procédés. Certains *candidats Monte Carlo* (section 2.3.5) démontrent qu'il est possible de réunir des conditions de fabrication conduisant à des fonctionnements fautifs de la part de la référence de tension *Chang*. Les essais 4 et 5 présentent plutôt des défaillances au niveau même du comportement en température, puisque certains des candidats démontrent qu'il est possible de recevoir des circuits qui ne produiront pas la tension de *Bandgap* à la sortie. Il est intéressant de noter qu'une configuration d'optimisation contenant plus de candidats Monte Carlo dans les analyses pourrait permettre de plus facilement écarter ces candidats fautifs des solutions explorées.

Dans la gamme des circuits restants (les non-fautifs), d'étonnantes synthèses sont rapportées. On trouve spécialement que les essais 3, 7, 8, 9 et 10 conjuguent, de manière simultanée, plusieurs très bonnes performances (Tableau 3.5). Pour résumer simplement, l'essai 3 s'oriente vers une faible consommation en courant; l'essai 7 privilégie plutôt un temps de démarrage court; l'essai 8 combine faible consommation en courant et excellent PSRR; l'essai 9 présente une très bonne stabilité de sa tension de sortie en température;

enfin, l'essai 10 minimise spécifiquement la dispersion de sa tension de sortie face aux variations des procédés. Notons au passage que ces différents résultats de synthèse ont tous été exécutés avec exactement la même configuration d'optimisation, à l'exception de la valeur du *germe* produisant les nombres aléatoires.

À travers ces différents résultats d'optimisation, on découvre *certaines déductions* qui témoignent des compromis prenant effet lors de la synthèse paramétrique du circuit. Par exemple, on constante rapidement qu'il existe un compromis flagrant entre la consommation en courant du circuit et sa propension à démarrer rapidement. Ceci est essentiellement déterminé par les capacités du circuit qui nécessitent un temps plus long pour se charger étant donné l'apport réduit en courant. De plus, toutes les synthèses ont convergées vers une capacité nulle pour C2. La capacité C2 est une capacité de Miller dans le régulateur de tension. Elle est judicieusement retirée par l'optimisateur puisqu'elle n'opère pas convenablement son rôle stabilisateur dans la boucle de rétroaction. Enfin, on distingue aussi que le bruit est inversement proportionnel à la surface occupée. Le principal responsable du bruit intrinsèque est le bruit 1/f qui diminue normalement selon la surface occupée par les grilles des transistors MOS.

4.2 Discussion sur la nature probabiliste des succès de la recherche par l'algorithme génétique

Un des points faible de l'algorithme génétique face aux autres types d'algorithmes est sa nature probabiliste concernant la découverte de circuits optimaux.

Généralement, les AG appliqués à l'optimisation des CA connaissent 3 catégories de coût distincts : l'effort de l'évaluation (simulateur), le critère de terminaison et l'obtention statistique de circuits performants.

Lorsqu'on accepte de payer le prix du côté de la performance du simulateur, alors on opte généralement pour un simulateur plus précis (donc plus lent) avec des modèles plus performants (donc plus complexes).

Lorsque le temps de convergence ne nous préoccupe pas, il est possible d'augmenter la durée d'optimisation, par l'augmentation du nombre de générations avant l'arrêt de l'algorithme (critère de fin).

Lorsqu'enfin, notre centre d'intérêt est la probabilité de trouver un très bon circuit, les paramètres importants deviennent la taille de la population et le nombre de redémarrages de l'algorithme.

Dans ce travail, un compromis a été recherché autour de ces trois coûts. En effet, le simulateur ainsi que les modèles des composants utilisés étaient de qualité industrielle. Ainsi, le coût associé à la perfomance du simulateur était élevé. Quant au temps de convergence, une barrière a été fixée à 100 générations pour des contraintes de temps,

soit environ 24 heures par optimisation à raison de 10 secondes par évaluation et 100 individus par population. Enfin, concernant l'obtention statistique, nous avons réalisé 10 fois la même optimisation afin de valider la probabilité de convergence à de bons circuits (5 cas sur 10). Il est important de noter que des populations de seulement 100 individus sont généralement considérées comme faible par la communauté intéressée aux algorithmes évolutifs [50]. Il n'est pas rare de voir des populations s'étendre à plus de 10000 individus, tout dépendant de la dimensionnalité des problèmes adressés.

4.3 Discussion à propos de la génération de circuits monstreux

Parmi les effets pervers notés à propos de l'utilisation d'un tel logiciel d'optimisation est la production de *circuits monstreux*. Ainsi, un concepteur qui a uniquement le contrôle sur la topologie du circuit et les plages paramétriques permises à l'entrée de l'optimisateur peut être confronté à ce problème. Un circuit monstreux est une catégorie de circuits qui présente en simulation *toutes les caractéristiques d'un fonctionnement admissible*, mais qui dans la réalité ne fonctionne pas correctement. En ce sens, un exemple de circuit monstreux est le cas où un circuit est optimisé sans tenir compte de la surface occupée. Il est ainsi possible que, sans la prise en compte de cette contrainte, l'optimisation tende à fuir vers un circuit toujours de plus en plus gros afin de maximiser la performance sur le bruit. Les circuits monstreux sont une forme de hantise qui a poussé plusieurs concepteurs à produire des circuits en dérogeant le moins possible d'une règle d'or simple : *tous les transistors impliqués doivent être en saturation.* Une stratégie unilatérale comme celle-ci assure généralement un bon fonctionnement des CA

produits, face : aux variations des procédés, aux fluctuations de l'alimentation et à la température. Cependant, dans le monde d'aujourd'hui, de telles stratégies conservatrices ne sont plus vraiment encouragées à cause de la trop forte consommation en puissance associée aux circuits résultants. C'est pourquoi, un bon montage d'optimisation, faisant intervenir les *bonnes* analyses pour évaluer le circuit, est toujours de la plus grande importance.

Il faut être conscient que le processus d'optimisation présenté dans ce mémoire est entièrement fondé sur l'exactitude des résultats de sortie d'un simulateur. Si celui-ci ne représente pas de manière conforme ou satisfaisante à la réalité, à cause d'erreurs de résolution numérique ou de décentrage sur les modèles utilisés, alors le résultat de l'optimisation sera aussi nécessairement fautif ou biaisé.

4.4 Discussion à propos de l'analyse Monte Carlo

L'intérêt premier et immédiat, d'un tel optimisateur, est de pouvoir réaliser des optimisations en regard à la température et aux variations Monte Carlo. Un des grands enjeux, pour amenuiser l'impact des variations des procédés, est de faire intervenir l'analyse Monte Carlo directement dans les analyses qui évaluent le circuit au cours de l'optimisation. Nous avons pu constaté qu'il est, de manière mesurable, possible de poursuivre une minimisation générale sur les performances tout en réduisant la dispersion des propriétés du circuit. Cela veut essentiellement dire que certains effets compensatoires peuvent trouver naissance dans l'optimisation. Des effets compensatoires qui se produisent au niveau même du point d'opération des transistors et qui les rendent

moins assujetties, globalement, aux variations des procédés. La compréhension complète de cet état de fait reste, en soi, transcendante. Mais son contrôle est assuré par une analyse qui vérifie cet impact et elle accompagne le concepteur lorsque celui-ci formule le désir de la minimiser.

4.5 Avancement scientifique apporté par l'outil proposé

Plusieurs outils d'optimisation de CA ont vu le jour, comme le démontre le grand nombre d'articles à ce sujet dans la littérature. De plus, l'application des algorithmes génétiques comme solveur à ce problème de dimensionnement n'est pas nouveau dans le domaine [9][107]. Cependant, là où cet outil se démarque, c'est sur sa flexibilité au niveau de la direction de manière interactive de l'optimisation, de la configuration de manière graphique de l'optimisation et de l'aiguillage de l'optimisation vers des buts précis grâce aux équations de type libre. La concaténation de ces différents avantages font de ce programme un outil unique en son genre.

Bien sûr, il faut savoir que la liberté de contrôle ajoutée par les équations libres au niveau des fonctions de *pointage* et *d'utilité* implique une certaine formation de la part de l'utilisateur de ce logiciel. Utilisé depuis près de deux ans chez **LTRIM Technologies Inc.**, cet outil supporte déjà plus de 4 utilisateurs connaissant adéquatement l'utilisation de AGO. Habituellement, ce logiciel permet aux concepteurs de produire des circuits beaucoup plus rapidement, beaucoup plus efficacement et ce en adressant des domaines de performance plus élevés.

Un des grands intérêts des logiciels d'optimisation est la possibilité de valider, en un temps relativement court, la viabilité d'une topologie de circuit trouvée par exemple dans un article scientifique ou imaginé par un concepteur de circuit.

Nous présentons aux points suivants, un sommaire récapitulatif de l'emploi pratique de l'outil d'optimisation analogique AGO à travers différentes stratégies et utilisations explorées :

- Optimisation de plusieurs amplificateurs opérationnels [100][42];
- Optimisation de plusieurs références de tension [73][59][100][42];
- Optimisation de plusieurs références de courant [100][42];
- Régression sur les paramètres technologiques du modèle d'un BJT par rapport à des mesures expérimentales.

L'investissement qui se pose généralement est celui-ci : vaut-il mieux performer une optimisation du circuit *à la main*, sans pour autant avoir de garantie sur l'obtention d'une solution acceptable en un temps respectable? Ou bien, est-il préférable de passer quelques jours pour configurer une optimisation et la laisser s'effectuer automatiquement par la suite?

Conclusion

Nous avons décrit dans ce mémoire une méthodologie d'optimisation appliquée à la synthèse de CA pour la réalisation d'ASIC.

Nous avons démontré que l'utilisation combinée d'outils de simulation et d'optimisation peut permettre aux concepteurs de rencontrer plus rapidement les performances désirées d'un CA.

Ce mémoire montre qu'il est possible à l'aide d'un algorithme génétique de réaliser cette tâche, et de déterminer les paramètres de conception optimaux afin de rencontrer des objectifs de performance, fixés par le concepteur, à atteindre.

Nous avons été en mesure de réaliser des optimisations qui tenaient compte de manière simultanée des multiples interdépendances entre les performances pour les différents domaines d'analyses (fréquentiel, paramétrique, transitoire).

Nous avons démontré que l'approche par optimisation peut se révéler être un outil puissant pour la conception de cellules analogiques. Ce système est présentement utilisé par des concepteurs de circuits industriels afin d'augmenter les performances des circuits et ce, dans un processus continu et entretenu. Des développements sont en cours afin de rendre cet outil toujours plus flexible et efficace.

Nous avons de plus énoncé qu'il était nécessaire d'améliorer l'analyse Monte Carlo qui décrit les perturbations aléatoires afin de pouvoir en tenir compte dans le processus d'optimisation des impacts des variations des procédés.

Nous avons enfin observé que les AG sont en mesure de rivaliser avec les autres techniques d'optimisation de CA. Par contre, la nature intrinsèquement heuristique de cet algorithme le prédispose à une variabilité sur la convergence lorsque le nombre de générations ainsi que la taille de la population sont faibles.

Bien sûr, malgré tous les avancements réalisés dans ce mémoire, il subsiste une liste notoire d'améliorations à apporter à cet outil :

- Sur le processus de simulation des circuits
  - Distribuer ce logiciel sur une ferme de calcul [55][83][54];
  - Accélérer le processus d'évaluation en encapsulant le simulateur [55];
- Sur l'algorithme
  - Utilisation d'une représentation génomique ajustable en cours d'optimisation (d'un pas grossier vers un pas fin et réajustement automatique des bornes);
  - Utiliser, de manière systématique, en conjoncture avec une méthode d'optimisation par descente de gradient afin d'accélérer la convergence;
- Améliorations à long terme
  - Entraîner un réseau de neurones pouvant converger vers la prédiction des candidats handicapés avant leur évaluation exhaustive par le simulateur de circuit;
  - Réaliser le placement et routage des dessins des masques directement dans le processus d'optimisation [49];

- Intégrer l'altération topologique libre au sein d'une optimisation [50].

Voici quelques points qui plaident en faveur d'une approche par optimisation paramétrique pour la synthèse des CA :

- La puissance de calcul toujours grandissante chez les ordinateurs;
- Le prix des ordinateurs qui diminue de jours en jours; selon un horizon prévisible (loi de Moore);
- Des simulateurs logiciels, ainsi que les modèles, qui vont toujours en se perfectionnant;
- Des technologies microélectroniques qui diminuent progressivement et des composantes électroniques qui présentent des comportements électriques de plus en plus non-linéaires;
- Des exigences toujours de plus en plus élevés en matière de performances associées aux CA;
- Des échéanciers alloués à la conception analogique toujours de plus en plus compressés;
- Une tendance générale de l'industrie d'adopter des outils de type WYSIWYG[27].

De nos jours, il est spectaculaire de constater les folles percées dans le domaine de la *synthèse analogique automatisée*. La conception analogique a longtemps été considérée comme un *art* que seul des artistes (concepteurs) pouvaient être en mesure

---

[27] What You See Is What You Get

d'effectuer. La nature intrinsèque très complexe des CA ne fait aucun doute. Pourtant aujourd'hui, on voit se profiler à l'horizon un nombre grandissant de profanes qui adressent ce domaine normalement restreint et réservé aux artistes de l'analogique.

En conclusion, la puissance de calcul grandissante des ordinateurs des décennies à venir saura changer le paradigme de l'*artiste analogue* afin de reléguer celui-ci au plan de l'*analyste des compromis* plutôt qu'à celui de la *recherche abrutissante*.

Bibliographie

[1] AGUIRRE, M.A., CHAVEZ, J., TORRALBA, A., FRANQUELO, L.G. (1994). Analog Design Optimization by means of a Tabu Search Approach, IEEE International Symposium on Circuits and Systems, ISCAS '94, 1, 375-378.

[2] AGUIRRE, M.A., CHAVEZ, J., TORRALBA, A., FRANQUELO, L.G., Sizing of Analog Cells by means of a Tabu Search Approach, non-publié.

[3] ALLEN, P.E., MACALUSO, E.R. (1985). AIDE2 : An Automated Analog IC Design System, Proceedings of IEEE Custom Integrated Circuits Conference, 498-501.

[4] ANDRE, D., BENNETT III, F.H., KOZA, J., KEANE, M.A., On the Theory of Designing Circuits using Genetic Programming and a Minimum of Domain Knowledge, Proceedings of the 1998 IEEE International Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence, 130-135.

[5] ANTAO, B.A.A., BRODERSEN, A.J., Techniques for Synthesis of Analog Integrated Circuits, IEEE Design & Test of Computers, 9, no.1, 8-18.

[6] ANTREICH, K., ECKMULLER, J., GRAEB, H., PRONATH, M., SCHENKEL, F., SCHWENCKER, R., ZIZALA, S. (2000). WiCkeD : Analog Circuit Synthesis Incorporating Mismatch, Proceedings of the IEEE 2000 Custom Integrated Circuits Conference, CICC 2000, 511-514.

[7] BATTITI, R., TECCHIOLLI, G. (1994). Simulated annealing and tabu search in the long run: a comparison on qap tasks, Computer and Mathematics with Applications, 28, no.6, 1-8.

[8] BAZARAA, M.S., SHETTY, C.M. (1979). Nonlinear Programming: Theory and Algorithms, John Wiley, New York.

[9] BENNETT III, F.H., KOZA, J.R., KEANE, M.A., MYDLOWEC, W., YU, J., STIFFELMAN, O. (1999). Evolution by Means of Genetic Programming of Analog Circuits that Perform Digital Functions, Proceedings of the Genetic and Evolutionary Computation Conference, 2, 1477-1483.

[10] BOYD, S. (2000). New approaches speed up optimization of analog designs, Electronic Design, 48, 62.

[11] BOYD, S., VANDENBERGHE, L. (1997). Introduction to Convex Optimization with Engineering Applications, Standford University, URL : http://www.stanford.edu/class/ee364/

[12] BOYD, S., VANDENBERGHE, L., GRANT, M. (1994). Efficient Convex Optimization for Engineering Design, Proceedings IFAC Symposium on Robust Control Design, 14-23.

[13] CARLEY, L.R., RUTENBAR, R.A., (1988). How to Automate Analog IC Designs, IEEE Spectrum, 25, no.8, 26-30.

[14] CHADHA, R., et al. (1987). WATOPT : An Optimizer for Circuit Applications, IEEE Transaction on Computer-Aided Design, CAD-6, 472-479.

[15] CONN, A.R., COULMAN, P.K., HARING, R.A., MORRILL, G.L., VISWESWARIAH, C., WU, C.W. (1998). JiffyTune : Circuit Optimization using time-domain sensitivity, IEEE Transactions on Computer-Aided Design, 17, 1292-1309.

[16] CONN, A.R., GOULD, N.I.M., TOINT, PH.L. (1989). Global convergence of a class of trust region algorithms for optimization with simple bounds, SIAM Journal on Numerical Analysis, 26, 764-767.

[17] DAGONIS, K., WALSH, K., LINARDES, P. (1989). Synthesis of Analog ASIC's using Optimization in Conjunction with Circuit Simulation Techniques, Proceedings. Second Annual IEEE ASIC Seminar and Exhibit, 10.3.1-10.3.5.

[18] DAOUD, Z., SPANOS, C.J. (1994). DORIC : Design of Optimal & Robust Integrated Circuits, Proceedings of IEEE Custom Integrated Circuit Conference, 361-364.

[19] DAWSON, J.L., BOYD, S.P., HERSHENSON, M.M., LEE, T.H. (2001). Optimal Allocation of Local Feedback in Multistage Amplifiers via Geometric Programming, IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, 48, 1-11.

[20] DEGRAUWE, M.G.R., NYS, O., DIJKSTRA, J., BITZ, S., GOFFART, B., VITTOZ, E., CSERVENY, S., MEIXENBERGER, C., STAPPEN, G.V.D., OGUEY, H.J. (1987). IDAC : An interactive Design Tool for Analog CMOS Circuits, IEEE Journal of Solid State Circuits, SC-22, 1106-1115.

[21] DEVAL, Y., BEGUERET, J.-B., TOMAS, J., FOUILLAT, P. (2000). Toward Analog Circuit Synthesis : A Global Methodology based upon Design Of Experiment, Proceedings. 13th Symposium on Integrated Circuits and Systems Design, 295-300.

[22] DHANWADA, N.R., NUNEZ-ALDANA, A., VEMURI, R., (1999). A Genetic Approch to Simultaneous Parameter Space Exploration and Constraint Transformation in Analog Synthesis, Proceedings of the 1999 IEEE International Symposium on Circuits and Systems, ISCAS '99, 6, 362-365.

[23] DIRECTOR, S., MALY, W., STROJWAS, A. (1990). VLSI Design for Manufacturing : Yield Enhancement, Kluwer Academic Publishers, USA.

[24] EL-TURKY, F., PERRY, E.E. (1989). BLADES : An Artificial intelligence Approach to Analog Circuit Design, IEEE Transactions on Computer-Aided Design, 8, 680-692.

[25] FISHBURN, J.P., DUNLOP, A.E. (1985). TILOS : A Posynomial Programming Approach to Transistor Sizing, IEEE International Conference on Computer-Aided Design, 326-328.

[26] GASS, S.I. (1958). Linear Programming: METHODS AND APPLICATIONS, McGraw-Hill, New York, 223 pages.

[27] GEYSKENS, P., DER KIUREGHIAN, A., DE ROECK, G. (1992). SORM analysis using Quasi-Newton optimization, First International Conference on Computational Stochastic Methods, Elsevier Applied Science, London, New York.

[28] GIELEN, G.G.E., RUTENBAR, R.A. (2000). Computer-Aided Design of Analog and Mixed-Signal Integrated Circuits, Proceedings of the IEEE, 88, no.12, 1825-1854.

[29] GIELEN, G.G.E., WALSCHARTS, H.C.C., SANSEN, W.M.C. (1989). ISAAC : A symbolic simulator for analog integrated circuits, IEEE Journal of Solid State Circuits, 24, no.6, 1587-1596.

[30] GIELEN, G.G.E., WALSCHARTS, H.C.C., SANSEN, W.M.C. (1990). Analog Circuit Design Optimization based on Symbolic Simulation and Simulated Annealing, IEEE Journal of Solid State Circuits, 25, no.3, 707-713.

[31] GLOVER, F. (1997). *Heuristics for Integer Programming using Surrogate Constraints*, Decision Sciences, 8, no.1, 156-166.

[32] GOFFART, B., MENEVAUT, L., MEIXENBERGER, C., DEGRAUWE, M. (1992). A Schematic Driven Synthesis Tool for Analog Circuits, Proceedings. Euro ASIC '92, 131-134.

[33] GOH, C., LI, Y. (2001). GA Automated Design and Synthesis of Analog Circuits with Practical Constraints, Proceedings of the 2001 Congress on Evolutionary Computation, 1, 170-177.

[34] GUPTA, S.K., HASAN, M.M. (1996). KANSYS : A CAD Tool for Analog Circuit Synthesis, Proceedings of 9th International Conference on VLSI Design, 333-334.

[35] HANSEN, M.P. (1997). Tabu Search for Multiobjective Optimization : MOTS, Proceedings of the 13th International Conference on Multiple Criteria Decision Making, MCDM'97, 16 pages.

[36] HARJANI, R., RUTENBAR, R.A., CARLEY, L.R. (1989). OASYS : A framework for analog circuit synthesis, IEEE Transactions on Computer-Aided Design, 8, 1247-1265.

[37] HARVEY, J.P., ELMASRY, M.I., LEUNG, B. (1992). STAIC : An Interactive Framework for Synthesizing CMOS and BiCMOS Analog Circuits, IEEE Transactions on Computer-Aided Design, 11, no.11, 1402-1417.

[38] HERSHENSON, M.M., BOYD, S.P., LEE, T.H. (1998). GPCAD : A Tool for CMOS Op-Amp Synthesis, 1998 IEEE/ACM International Conference on Computer-Aided Design, ICCAD 98, 296-303.

[39] HERSHENSON, M.M., BOYD, S.P., LEE, T.H. (2001). Optimal Design of a CMOS Op-Amp via Geometric Programming, IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems, 20, no.1, 1-21.

[40] HOLLAND, J.H. (1975). Adaptation in Natural and Artificial Systems, Michigan Press. University, Michigan.

[41] HOLMAN, W. (1994). Low Noise CMOS Voltage Reference, Thèse de doctorat, Georgia Institute of Technology.

[42] JOHNS, D.A., MARTIN, K. (1997). Analog Integrated Circuit Design, John Wiley & Son, University of Toronto, 706 pages.

[43] KERAMAT, M., KIELBASA, R. (1998). OPTOMEGA : An Environment for Analog Circuit Optimization, Proceedings of the 1998 IEEE International Symposium on Circuits and Systems, ISCAS '98, 6, 122-125.

[44] KIRKPATRICK, S., GELATT, C.D., VECCHI, M.P.J. (1983). Optimisation by Simulated Annealing, Science, 220, 671-680.

[45] KOH, H.Y., SÉQUIN, C.H., GRAY, P.R. (1990). OPASYN : A compiler for CMOS operational amplifiers, IEEE Transactions on Computer-Aided Design, 9, 113-125.

[46] KORTANEK, K.O., XU, X., YE, Y. (1996). An infeasible interior-point algorithm for solving primal and dual geometric programs, Math Programming, 76, 155-181.

[47] KOZA, J.R. (1995). Survey of Genetic Algorithms and Genetic Programming, Conference record WESCON/'95, Microelectronics Communications Technology Producing Quality Products Mobile and Portable Power Emerging Technologies, 589-594.

[48] KOZA, J.R. (1999). Darwinian Invention and Problem Solving by means of Genetic Programming, Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics, SMC '99, 3, 604-609.

[49] KOZA, J.R., BENNETH III, F.H. (1999). Automatic Synthesis, Placement, and Routing of Electrical Circuits by Means of Genetic Programming, Advances in

Genetic Programming Volume III, MIT Press, Massachusetts, USA, 105-134, 476 pages.

[50] KOZA, J.R., BENNETH III, F.H., ANDRE, D., KEANE, M.A., (1999). Genetic Programming III : Darwinian Invention and Problem Solving, Morgan Kaufmann Publisher, USA, 1154 pages.

[51] KOZA, J.R., BENNETH III, F.H., ANDRE, D., KEANE, M.A., DUNLAP, F. (1997). Automated Synthesis of Analog Electrical Circuits by means of Genetic Programming, IEEE Transactions on Evolutionary Computation, 1, no.2, 109-128.

[52] KOZA, J.R., BENNETT III, F.H., ANDRE, D., KEANE, M.A. (1996). Four Problems for which a Computer Program Evolved by Genetic Programming is Competitive with Human Performance, Proceedings of IEEE International Conference on Evolutionary Computation, 1-10.

[53] KOZA, J.R., BENNETT III, F.H., MYDLOWEC, W., KEANE, M.A., YU, J., STIFFELMAN, O. (1999). Search for the Impossible using Genetic Programming, Proceedings of the Genetic and Evolutionary Computation Conference, 2, 1083-1091.

[54] KRASNICKI, M., PHELPS, R., RUTENBAR, R., CARLEY, L.R. (1999). MAELSTROM : Efficient Simulation-Based Synthesis for Custom Analog Cells, Proceedings of 36th ACM/IEEE Design Automation Conference, 945-950,.

[55] KRASNICKI, M.J., PHELPS, R., HELLUMS, J.R., MCCLUNG, M., RUTENBAR, R.A., CARLEY, L.R. (2001). ASF : A Practical Simulation-Based Methodology for the Synthesis of Custom Analog Circuits, IEEE/ACM International Conference on Computer Aided Design, ICCAD 2001, 350-357.

[56] KRUISKAMP, W., LEENAERTS, D. (1995). DARWIN : CMOS op amp synthesis by means of a genetic algorithm, Proceedings of the 32nd Annual Design Automation Conference, 433-438,.

[57] LAI, J.C., KUENG, J.S., CHEN, H.C., FERNANDEZ, F.J. (1988). ADOPT : A CAD Tool for Analog Circuit Design, IEEE Circuits and Devices Magazine, 29-30.

[58] LEE, C.-H., CORNISH, J., MCCLELLAN, K., CHOMA, J.JR. (1998). Design of Low Jitter PLL for Clock Generator with Supply Noise Insensitive VCO, Proceedings of the 1998 IEEE International Symposium on Circuits and Systems, ISCAS '98, 1, 233-236.

[59] LEE, C.-H., MCCLELLAN, K., CHOMA, J.JR. (2001). A Supply-Noise-Insensitive CMOS PLL with a Voltage Regulator using DC-DC Capacitive Converter, IEEE Journal of Solid State Circuits, 36, 1453-1463.

[60] LEENAERTS, D. (1991). TOPICS : A New Hierarchical Design Tool using an Expert System and Interval Analysis, Proceedings of ESSCIRC, 37-40.

[61] LEVENBERG, K. (1944). A method for the solution of certain problems in least squares, Quart. Appl. Math., 2, 164-168.

[62] LOHN, J.D., COLOMBANO, S.P. (1999). A Circuit Representation Technique for Automated Circuit Design, IEEE Transactions on Evolutionary Computation, 3, no.3, 205-219.

[63] MANDAL, P., VISVANATHAN, V. (2001). CMOS Op-Amp Sizing using Geometric Programming Formulation, IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems, 20, no.1, 22-38.

[64] MARIN, D., ESCUDERO, J., OLIVER, J., FLANDRE, D. (1997). CONNAN: A Computer-Aided Design Tool based on Sharing Information for Analog Circuit Sizing, Universitat Autònoma de Barcelona PATMOS'97, Seventh International Workshop Program, Université Catholique de Louvain, Belgique.

[65] MARQUARDT, D. (1963). An algorithm for least-squares estimation of nonlinear parameters, SIAM Journal of Applied Mathematics, 11, 431-441,.

[66] MASTERS, T. (1995). Advanced Algorithms for Neural Networks: A C++ Sourcebook, John Wiley and Sons, New York, ISBN: 0-471-10588-0.

[67] MAYARAM, K., YANG, P., CHERN, J.-H., BURCH, R., ARLEDGE, L., COX, P. (1990). A Parallel Block-Diagonal Preconditioned Conjugate-Gradient Solution Algorithm for Circuit and Device Simulations, IEEE International Conference on Computer-Aided Design, ICCAD'90, 446-449.

[68] MEDEIRO, F., DOMÍNGUEZ-CASTRO, R., RODRÍGUEZ, A., HUERTAS, J.L. (1992). A Prototype Tool for Optimum Analog Sizing using Simulated Annealing, Proceedings of IEEE International Symposium on Circuits & Systems, 1933-1936.

[69] MEDEIRO, F., FERNANDEZ, F.V., DOMINGUEZ-CASTRO, R., RODRIGUEZ-VAZQUEZ, A. (1994). A Statistical Optimization-Based Approach for Automated Sizing of Analog Cells, Proceedings of ACM/IEEE International Conference on Computer-Aided Design, 594-597.

[70] MOUNIR, F., BOZENA, K. (1995). FPAD : A Fuzzy Nonlinear Programming Approach to Analog Circuit Design, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 14, no.7, 785-793.

[71] NAGEL, L.W. (1985). SPICE2 : A computer program to simulate semiconductor circuits, Electronics Research Laboratory, Université de Californie, Berkeley, Memo ERL-M520.

[72] NAM, D., SEO, Y.-D., PARK, L.-J., PARK, C.-H., KIM, B. (1998). Parameter Optimization of a Voltage Reference Circuit Using EP, Proceedings of the IEEE International Conference on Evolutionary Computation 1998, IEEE World Congress on Computational Intelligence, 301-305.

[73] NAM, D., SEO, Y.-D., PARK, L.-J., PARK, C.-H., KIM, B. (2001). Parameter Optimization of an On-Chip Voltage Reference Circuit Using Evolutionary Programming, IEEE Transactions on Evolutionary Computation, 5, no.4, 414-421.

[74] NELDER, J.A., MEAD, R. (1965). A Simplex Method for Function Minimization, Computer Journal, 7, 308-313.

[75] NGUYEN, T.V., FELDMANN, P., DIRECTOR, S.W., ROHTER, R.A. (1989). *SPECS simulation validation with efficient transient sensitivity computation*, IEEE International Conference on Computer-Aided Design, 252-255.

[76] NING, Z., MOUTHAAN, T., WALLINGA, H. (1991). SEAS : A Simulated Evolution Approach for Analog Circuit Synthesis, Proceedings of IEEE Custom Integrated Circuit Conference, 5.2.1-5.2.4.

[77] NYE, W., RILEY, D.C., SANGIOOVANNI-VINCENTELLI, A., TITS, A.L. (1988). DELIGHT.SPICE : An Optimization-Based System for the Design of Integrated Circuits, IEEE Transactions on Computer-Aided Design, 7, 501-518.

[78] NYE, W.T., POLAK, E., SANGIOVANNI-VINCENTELLI, A. (1981). DELIGHT : An Optimization-Based Computer-Aided Design System, Proceedings of IEEE International Symposium on Circuit & Systems, 851-855.

[79] OCHOTTA, E.S., RUTENBAR, R.A., CARLEY, L.R. (1996). Synthesis of high-performance Analog Circuits in ASTRX/OBLX, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 15, no.3, 273-293.

[80] ODONERA, H., KANBARA, H., TAMARU, K. (1990). Operational-Amplifier Compilation with Performance Optimization, IEEE Journal of Solid State Circuits, 25, 466-473.

[81] PARETO, V. (1896). Cours d'Économie Politique, Rouge, Lausanne, Suisse.

[82] PELLERIN, D., TAYLOR, D. (1996). VHDL Made Easy!, Upper Saddle River, New Jersey, Prentice-Hall, 419 pages.

[83] PHELPS, R., KRASNICKI, M., RUTENBAR, R., CARLEY, L.R., HELLUMS, J.R. (2000). ANACONDA : Simulation-Based Synthesis of Analog Circuits via Stochastic Pattern Search, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 19, no.6, 703-717.

[84] PLAS, G.V.D., DEBYSER, G., LEYN, F., LAMPAERT, K., VANDENBUSSCHE, J., GIELEN, G.G.E., SANSEN, W., VESELINOVIC, P., LEENAERT, D. (2001). AMGIE : A Synthesis Environment for CMOS Analog Integrated Circuits, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 20, no.9, 1037-1058.

[85] RAZIK, H., DEFRANOUX, C., REZZOUG, A. (2000). Identification of induction motor using a genetic algorithm and a quasi-newton algorithm, VII IEEE International Power Electronics Congress, CIEP 2000, 65-70.

[86] RICHMAN, B.A., WINDLEY, P.J. (1993). ACAD: A Hierarchical Approach to CMOS Design Analysis, Proceedings of the IEEE 1993 Custom Integrated Circuits Conference, 14.6.1-14.6.4.

[87] ROSENTHAL, R.E. (1985). Principles of Multiobjective Optimization, Decision Sciences, 16, 133-152.

[88] RUTENBAR, R.A. (1989). Simulated Annealing Algorithm : An Overview, IEEE Circuits and Devices Magazine, 5, no.1, 19-26.

[89] RUTENBAR, R.A. (1993). Analog Design Automation : Where are we? Where are we going?, Proceedings of the IEEE Custom Integrated Circuits Conference, 13.1.1-13.1.7.

[90] SEDGEWICK, R. (1990). Algorithms in C, Addison-Wesley, Princeton University, 657 pages.

[91] SEO, Y.-D., NAM, D., YOON, B.-J., CHOI, I.-H., KIM, B. (1997). Low-Power CMOS On-Chip Voltage Reference Using MOS PTAT : An EP Approach, Proceedings. of the Tenth Annual IEEE International ASIC Conference and Exhibit, 316-320.

[92] SHYU, J., SANGIOOVANNI-VINCENTELLI, A. (1988). ECSTASY : A New Environment for IC Design Optimization, IEEE International Conference on Computer-Aided Design, 484-487.

[93] SWINGS, K., DONNAY, S., SANSEN, W. (1991). HECTOR : A Hierarchical Topology-Construction Program for Analog Circuits based on a Declarative Approach to Circuit Modeling, Proceedings of the IEEE 1991 Custom Integrated Circuits Conference, 5.3.1-5.3.4.

[94] SWINGS, K., SANSEN, W.M.C. (1993). ARIADNE: A Constrained-Based Approach to Computer-Aided Synthesis and Modeling of Analog integrated Circuits, Analog Integrated Circuits and Signal Processing, 3, no.3, 197–216.

[95] TORRALBA, A., CHAVEZ, J., FRANQUELO, L.G. (1996). FASY : A Fuzzy-Logic Based Tool for Analog Synthesis, IEEE Transaction on Computer-Aided Design, 15, 705-715.

[96] TRONTELJ, J., TRONTELJ, L., PLETERŠEK, A., VODOPIVEC, A. (1990). Rule Based CAD Tools for Analog Circuit Synthesis and Layout Compilation, IEEE Custom Integrated Circuits Conference, 14.8.1-14.8.4.

[97] VAN LAARHOVEN, P.J.M., AARTS, E.H.L. (1987). Simulated Annealing : Theory and Applications, Amsterdam, The Netherlands, Reidel.

[98] VAN VELDHUIZEN, D.A., LAMONT, G.B. (2000). Multiobjective Evolutionary Algorithms : Analyzing the State-of-the-Art, Evolutionary Computation, 8, 125-147.

[99] VANCORENLAND, P., DE RANTER, C., STEYAERT, M., GIELEN, G. (2000). Optimal RF design using Smart Evolutionary Algorithms, Proceedings on Design Automation Conference, 7-10.

[100] VITTOZ, E., FELLRATH, J. (1977). CMOS analog integrated circuits based on weak inversion operation, IEEE Journal of Solid State Circuits, SC-12, 224-231.

[101] WALL, M. (2000). GALib: Genetic Algorithm Library, version: 2.4.5, MIT, mars, URL : http://lancet.mit.edu/ga/

[102] WÓJCIKOWSKI, M., GLINIANOWICZ, J., BIALKO, M. (1996). System for Optimization of Electronic Circuits using Genetic Algorithm, Proceedings of IEEE International Conference Electronics Circuit & Systems, 247-250.

[103] WONG, D.F., LEONG, H.W., LIU, C.L. (1988). Simulated Annealing for VLSI Design, Kluwer, Norwell, MA.

[104] WRIGHT, S.J. (1997). Primal-Dual interior-Point Methods, SIAM, Philadelphia.

[105] YANG, H.Z., FAN, C.Z., WANG, H., LIU, R.S. (1996). Simulated Annealing Algorithm with Multi-Molecule : an Approch to Analog Synthesis, Proceedings of European Design and Test Conference, 571-575.

[106] ZEBULUM, R.S., PACHECO, M.A., VELLASCO, M. (1998). A Multi-Objective Optimisation Methodology Applied to the Synthesis of Low-Power Operational Amplifier, Proceedings of the XIII international conference on microelectronis and packaging, The brazilian Microelectronics Society, 1, 246-271.

[107] ZEBULUM, R.S., PACHECO, M.A., VELLASCO, M. (1998). Synthesis of CMOS Operational Amplifiers Through Genetic Algorithms, Proceedings of the XI Brazilian Symposium on Integrated Circuit Design, Rio de Janeiro, 125-128.

[108] "_____", (2002). HSpice User Manual, Version: 2001.4, Synopsys, 2000 pages.

[109] "_____", (2003). PSpice, Cadence Corporate, URL : http://www.cadencepcb.com/products/ pspice/datasheets/default.asp

[110] "_____", (2003). SmartSpice DataSheet, Silvaco, URL : http://www.silvaco.com/products /analog/crusade/smartspice/smartspice_br.html

[111] "_____", (2003). T-Spice, TannerEDA, URL : http://www.tanner.com/

Annexe I. Netlists

Listage 4.1 Description *netlist* de la référence de tension avec contraintes structurelles (Chang.sp).

```
*Référence de tension Chang avec appariements

.prot
.lib 'Chang.lib' PARAM
.lib 'Chang.lib' OPTIONS
.lib 'lt035tsmc.lib' MC_REEL
.unprot

$Définition des paramètres optimisables
.param l1= opt(1.00u, 1u,  5u,0.5u)
.param l2=l1
.param l3=l1
.param l4= opt(1.00u, 1u,  5u,0.5u)
.param l5=l4
.param l6= opt(1.00u, 1u,  5u,0.5u)
.param l7=l6
.param l8= opt(1.00u, 1u,  5u,0.5u)
.param l9=l8
.param l10=opt(1.00u, 1u,  5u,0.5u)
.param l11=opt(1.00u, 1u,  5u,0.5u)
.param l12=l11
.param l13=l4
.param l14=l8
.param l15=opt(1.00u, 1u,  5u,0.5u)
.param l16=opt(20.0u, 1u, 20u,1.0u)
.param l17=opt(20.0u, 1u, 20u,1.0u)
.param w=  opt(   5u, 5u, 20u,  5u)
.param m1= opt(    1,  1,  10,   1)
.param m2= opt(    1,  1,  10,   1)
.param m3= opt(    1,  1,  10,   1)
.param m4= opt(    1,  1,  10,   1)
.param m5= opt(    1,  1,  10,   1)
.param m6= opt(    1,  1,  10,   1)
.param m7=m6
.param m8= opt(    1,  1,  10,   1)
.param m9=m8
.param m10=opt(    1,  1,  10,   1)
.param m11=opt(    1,  1,  10,   1)
.param m12=opt(    1,  1,  10,   1)
.param m13=opt(    1,  1,  10,   1)
.param m14=opt(    1,  1,  10,   1)
.param m15=opt(    1,  1,  10,   1)
.param m16=opt(    1,  1,  10,   1)
.param m17=opt(    1,  1,  10,   1)
.param c1= opt( 0.0p, 0p, 10p,100f)
.param c2= opt( 0.0p, 0p, 10p,100f)
.param c3= opt( 0.0p, 0p, 10p,100f)
.param c4= opt( 0.0p, 0p, 50p,  1p)
.param lr1=opt( 500u,  0,  2m, 50u)
.param lr2=opt( 2.0m,  0,  4m, 50u)
.param lr3=opt( 500u,  0,  2m, 50u)
.param wr1=opt(   1u, 1u,  5u,  1u)
.param wr2=wr1
.param wr3=wr1
.param mq1=opt(   1,  1,   1,   1)
```

```
.param mq2=opt(   1,  1,   1,   1)
.param mq3=opt(   1,  1,  10,   1)
.param mq4=opt(   1,  1,   1,   1)

$Module générateur de bandgap
xM1 VBG N2  N1  N1   PCH  L=l1  W=w   Mx=m1
xM2 N2  N2  N1  N1   PCH  L=l2  W=w   Mx=m2
xM3 N3  N2  N1  N1   PCH  L=l3  W=w   Mx=m3
xM4 N3  N3  N5  GND  NCH  L=l4  W=w   Mx=m4
xM5 N2  N3  N6  GND  NCH  L=l5  W=w   Mx=m5
xR1 N6  N7           rpo2 L=lr1 W=wr1
xR2 VBG N4           rpo2 L=lr2 W=wr2
xQ1 GND GND N4       pnp10            Mx=mq1
xQ2 GND GND N5       pnp10            Mx=mq2
xQ3 GND GND N7       pnp10            Mx=mq3
C1 VBG GND c1

$Module régulateur d'alimentation et de courant
xM6  N8  N2  N1  N1  PCH  L=l6  W=w   Mx=m6
xM7  N9  N3  N1  N1  PCH  L=l7  W=w   Mx=m7
xM8  N8  N8  GND GND NCH  L=l8  W=w   Mx=m8
xM9  N9  N8  GND GND NCH  L=l9  W=w   Mx=m9
xM10 N1  N9  GND GND NCH  L=l10 W=w   Mx=m10
xM11 N11 N11 N10 N10 PCH  L=l11 W=w   Mx=m11
xM12 N1  N11 N10 N10 PCH  L=l12 W=w   Mx=m12
xM13 N11 N3  N12 GND NCH  L=l13 W=w   Mx=m13
xM14 N12 N8  GND GND NCH  L=l14 W=w   Mx=m14
C2  N1  N9  c2

$Filtre RC
xR3 VDD N10          rpo2 L=lr3 W=wr3
C4 N10 GND c4

$Module de démarrage
xM15 N2  N13 N3  GND NCH  L=l15 W=w   Mx=m15
xM16 N13 N13 VDD VDD PCH  L=l16 W=w   Mx=m16
xM17 N13 N13 N14 GND NCH  L=l17 W=w   Mx=m17
xQ4  GND GND N14     pnp10            Mx=mq4
C3  N3  GND c3

$Les analyses
.LIB 'Chang.lib' TEMPCO_VDD
.alter
.DEL LIB 'Chang.lib' TEMPCO_VDD

.LIB 'Chang.lib' TEMPCO_MC
.alter
.DEL LIB 'Chang.lib' TEMPCO_MC

.LIB 'Chang.lib' PSRR_TEMP
.alter
.DEL LIB 'Chang.lib' PSRR_TEMP

.LIB 'Chang.lib' BOOT_MC
$.alter
$.DEL LIB 'Chang.lib' BOOT_MC
.END
```

Listage 4.2 Bibliothèque des analyses pour la référence de tension lors de l'optimisation (Chang.lib).

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$Sections des librairies$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
*TempCo @ plage VDD (+/- 10%) w/ nominal process
*Courant @ plage VDD (+/- 10%) w/ nominal process
*Area
.LIB TEMPCO_VDD
 .LIB 'Chang.lib' AREA       $ Évaluation de la surface occupée

 Valim VDD GND DC alim
 .DC SWEEP temp LIN temp_pas temp_min temp_max
 +   SWEEP alim LIN vdd_pas  vdd_min  vdd_max
 .PROBE 'vbg'=par('v(vbg)')
 .PROBE 'courant'=par('abs(i(Valim))')
 .PROBE 'aire'=par('abs(i(Varea))')
 .PROBE 'iq1'=par('abs(i3(Q1))')
 .PROBE 'iq2'=par('abs(i3(Q2))')
 .PROBE 'iq3'=par('abs(i3(Q3))')
.ENDL TEMPCO_VDD

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
*TempCo @ process w/ VDD=+3.3V
*Courant @ process w/ VDD=+3.3V
.LIB TEMPCO_MC
 Valim VDD GND DC alim
 .DC SWEEP temp LIN temp_pas temp_min temp_max
 +   SWEEP MONTE=5
 .PROBE 'vbg'=par('v(vbg)')
 .PROBE 'courant'=par('abs(i(Valim))')
.ENDL TEMPCO_MC

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
*PSRR @ temperature w/ VDD=+3.3V
*Bruit @ temperature
.LIB PSRR_TEMP
 Valim VDD GND DC alim AC 1
 .AC DEC freq_pas freq_min freq_max
 +   SWEEP temp LIN temp_pas temp_min temp_max
 .PROBE 'magnitude_vbg'=par('vdb(vbg)')
 .PROBE 'phase_vbg'=par('vp(vbg)')

 .NOISE v(vbg) Valim 0    $* bruit reporté à la source
 .PROBE 'densite_spectrale_bruit'=par('onoise(m)')
 .MEASURE AC 'volt_noise_sq_10'  INTEG PAR('onoise(m)*onoise(m)') FROM=0.1 TO=10
 .MEASURE AC 'vnoise_rms_10'     PARAM='sqrt(volt_noise_sq_10)'
 .MEASURE AC 'volt_noise_sq_10k' INTEG PAR('onoise(m)*onoise(m)') FROM=10 TO=10k
 .MEASURE AC 'vnoise_rms_10k'    PARAM='sqrt(volt_noise_sq_10k)'
.ENDL PSRR_TEMP

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
*Boot @ process w/ VDD=10usec Ramp
.LIB BOOT_MC
 Valim Vdd gnd PWL('0us', 0V '10us' 0V '20u' alim)
 .TRAN 50u 4000us
 +SWEEP MONTE=5
 .probe V(vbg)
.ENDL BOOT_MC
```

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
.LIB AREA
 .param area_m1='2*(l1*w*m1)'
 .param area_m2='2*(l2*w*m2)'
 .param area_m3='2*(l3*w*m3)'
 .param area_m4='2*(l4*w*m4)'
 .param area_m5='2*(l5*w*m5)'
 .param area_m6='2*(l6*w*m6)'
 .param area_m7='2*(l7*w*m7)'
 .param area_m8='2*(l8*w*m8)'
 .param area_m9='2*(l9*w*m9)'
 .param area_m10='2*(l10*w*m10)'
 .param area_m11='2*(l11*w*m11)'
 .param area_m12='2*(l12*w*m12)'
 .param area_m13='2*(l13*w*m13)'
 .param area_m14='2*(l14*w*m14)'
 .param area_m15='2*(l15*w*m15)'
 .param area_m16='2*(l16*w*m16)'
 .param area_m17='2*(l17*w*m17)'

 .param area_r1='lr1*(wr1+2u)'
 .param area_r2='lr2*(wr2+2u)'
 .param area_r3='lr3*(wr3+2u)'

 .param area_c1='150e-6*1e-6*((c1-2.19e-15)/(133.5e-15))'
 .param area_c2='150e-6*1e-6*((c2-2.19e-15)/(133.5e-15))'
 .param area_c3='150e-6*1e-6*((c3-2.19e-15)/(133.5e-15))'
 .param area_c4='150e-6*1e-6*((c4-2.19e-15)/(133.5e-15))'

 .param area_q1='42u*42u*(mq1)'
 .param area_q2='42u*42u*(mq2)'
 .param area_q3='42u*42u*(mq3)'
 .param area_q4='42u*42u*(mq4)'

 .param area_m='area_m1+area_m2+area_m3+area_m4+area_m5\\
              +area_m6+area_m7+area_m8+area_m9+area_m10\\
              +area_m11+area_m12+area_m13+area_m14\\
              +area_m15+area_m16+area_m17'
 .param area_c='area_c1+area_c2+area_c3+area_c4'
 .param area_r='area_r1+area_r2+area_r3'
 .param area_q='area_q1+area_q2+area_q3+area_q4'

 .param area='area_m+area_c+area_r+area_q'

 VAREA AREA 0 'sqrt(area)'
 RAREA AREA 0 1
.ENDL AREA

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
.LIB OPTIONS
 .OPTIONS
$ Options pour les fichiers de sortie
 + POST        = 2    $fichier binaire
 + POST_VERSION       = 9601 $format 9007, 9601 (32-bit) ou 2001 (64-bit)
* + PROBE      = 1    $sortir seulement les résultats demandés
 + MEASOUT     = 1    $sortir les fichiers de mesure
 + MEASDGT     = 7    $précision des fichiers de mesure
 + INGOLD      = 0    $format de puissance littérale (pico=p)
$ Options pour la simulation transitoire
* + AUTOSTOP   = 1    $terminer la simulation transient après avoir réalisé toutes
                      $les mesures
 + ITL5        = 5000 $limite le nombre de points d'une analyse TR à 5000
* + INTERP     = 1    $réalise un échantillonage pour les résultats de sortie
$ Autres options
 + TNOM        = 0    $affectation de la température nominale à 0 degC
.ENDL OPTIONS
```

```
.LIB PARAM
 .param alim=3.3
 .param vdd_min='alim*(1-0.1)'       $+/1 10% autour de la tension d'alimentation
 .param vdd_max='alim*(1+0.1)'
 .param vdd_pas=5

 .param temp_min=-40
 .param temp_max=85
 .param temp_pas=7

 .param freq_min=0.1
 .param freq_max=10g
 .param freq_pas=3
 .ENDL PARAM
```

Annexe II. Fichier Monte Carlo (TSMC035)

Listage 4.3 Bibliothèque Monte Carlo du fichier technologique TSMC 0.35µm.

```
$V1.2  TECHNOLOGIE TSMC 0.35um Polycide MIXEDMODE 1P5M+ 3.3V/5.0V
$       w/ poly2cap.
$
$>>> [08/09/02]       Création de la librairie lt035tsmc.lib              (v1.0)
$>>> [31/10/02]       Correction MOS et RPOLY2                    (v1.1)
$>>> [31/10/02]       Retrait des contacts de RPOLY2                      (v1.2)
$>>> [31/10/02]       Ajout des régression pour les MOS           (v1.2)
$>>> [31/10/02]       Ajout de la dispersionpour les RPOLY2               (v1.2)

.LIB MC_REEL
 .OPTIONS PARHIER=LOCAL
$$$$$$$$$$$$$$$$$$$$$$$$$$$$DEBUT SECTION MOS$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
*************************************************************************
 .lib 'lt035tsmc.lib' MOS_ABS        $calcul les deviations abs. et
                                     $les inter-correlations directes
 .lib 'mm0355v.l'     MOS            $la definition des MOS
*************************************************************************
$$$$$$$$$$$$$$$$$$$$$$$$$$$FIN SECTION MOS$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

$$$$$$$$$$$$$$$$$$$$$$$$$$DEBUT SECTION BJT$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
*************************************************************************
 .lib 'lt035tsmc.lib' BJT_ABS
 .lib 'mm0355v.l'     BIP
*************************************************************************
$$$$$$$$$$$$$$$$$$$$$$$$$$$FIN SECTION BJT$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

$$$$$$$$$$$$$$$$$$$$$$$$$$DEBUT SECTION RES$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
 .param r_var_abs     = 0.10 $decalage absolue des proprietes abs. des
                             $res. agit surtout sur ptc1, pvc1 et pvc2
                             $rsh et dw ayant des grandeurs physiques
 .param r_var_rel     = 0.10 $mismatch rel. entre les lignes de res.
                             $agit sur rsh indirecte, ptc1, pvc1 et
                             $pvc2
 .param r_sigma_abs   = 3
 .param r_sigma_rel   = 3
 .param r_mult_abs    = 1
 .param r_mult_rel    = 1
*************************************************************************
 .lib 'lt035tsmc.lib' RES

*************************************************************************
$$$$$$$$$$$$$$$$$$$$$$$$$$$FIN SECTION RES$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
.ENDL MC_REEL

*************************************************************************
*************************************************************************
*************************************************************************
*************************************************************************
******                      SUBCKT & RELATIF                         *******
*************************************************************************
*************************************************************************
*************************************************************************

.LIB MOS

*************************************************************************
 .lib 'lt035tsmc.lib' MOS_PARAM      $defn des grandeurs physiques
```

```
 .param mos_rcn_abs    = gauss(mos_rcn,mos_var_rc%,m_sigma_abs,m_mult_abs)
 .param mos_rcp_abs    = gauss(mos_rcp,mos_var_rc%,m_sigma_abs,m_mult_abs)
*************************************************************************

 .subckt nch N1 N2 N3 N4 l=1u w=1u mx=1
  .param mos_vtn_rel  = '1e-3*(mos_dsp_An*1e-6/sqrt((l+(3E-
08+dxl)+dxl_rel)*(w+(0+dxw)+dxw_rel)*mx)+mos_dsp_Bn)'
  .param mos_dev_rel  = agauss(0,mos_vtn_rel,m_sigma_rel,m_mult_rel) $[-(mV),(mV)]
  .param dxl_rel      = agauss(0,mos_dsp_dxn,m_sigma_rel,m_mult_rel)
  .param dxw_rel      = agauss(0,mos_dsp_dxn,m_sigma_rel,m_mult_rel)
  .param rcn          = gauss(mos_rcn_abs,mos_dsp_rc%,m_sigma_rel,m_mult_rel)
  .param rsc          = 'abs(rcn) / max(1,int((w+0.1u)/0.8u))' $nb. de contact a la source
  .param rdc          = 'abs(rcn) / max(1,int((w+0.1u)/0.8u))' $nb. de contact au drain
  .param nrs          = '1.25u/max(0.7u,LV2(M1))' $nb. de carreau de diffusion a la source
  .param nrd          = '1.25u/max(0.7u,LV2(M1))' $nb. de carreau de diffusion au drain
  M1 N1 N2 N3 N4 nch L='max(0.35u,l+dxl_rel)' W='max(0.40u,w+dxw_rel)'
  + M='mx' DELVTO='mos_dev_rel'
  + NRS=nrs NRD=nrd RSC=rsc RDC=rdc  $les resistances de contact
 .ends

 .subckt pch N1 N2 N3 N4 l=1u w=1u mx=1
  .param mos_vtp_rel  = '1e-3*(mos_dsp_Ap*1e-6/sqrt((l+(3E-
08+dxl)+dxl_rel)*(w+(0+dxw)+dxw_rel)*mx)+mos_dsp_Bp)'
  .param mos_dev_rel  = agauss(0,mos_vtp_rel,m_sigma_rel,m_mult_rel)
  .param dxl_rel      = agauss(0,mos_dsp_dxp,m_sigma_rel,m_mult_rel)
  .param dxw_rel      = agauss(0,mos_dsp_dxp,m_sigma_rel,m_mult_rel)
  .param rcp          = gauss(mos_rcp_abs,mos_dsp_rc%,m_sigma_rel,m_mult_rel)
  .param rsc          = 'abs(rcp) / max(1,int((w+0.1u)/0.8u))'
  .param rdc          = 'abs(rcp) / max(1,int((w+0.1u)/0.8u))'
  .param nrs          = '1.25u/max(0.7u,LV2(M1))'
  .param nrd          = '1.25u/max(0.7u,LV2(M1))'
  M1 N1 N2 N3 N4 pch L='max(0.35u,l+dxl_rel)' W='max(0.40u,w+dxw_rel)'
  + M='mx' DELVTO='mos_dev_rel'
  + NRS=nrs NRD=nrd RSC=rsc RDC=rdc  $les resistances de contact
 .ends

.ENDL MOS

.LIB BJT
*************************************************************************
 .lib 'lt035tsmc.lib' BJT_PARAM
*************************************************************************

  .subckt pnp10 N1 N2 N3 mx=1
   .param bjt_dev_rel  = gauss(1,bjt_var_rel,'q_sigma_rel*sqrt(mx*ka_pnp10)',q_mult_rel)
   Q1 N1 N2 N3 pnp10 AREA='bjt_dev_rel' M='mx'
  .ends

.ENDL BJT

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$  RES LIBRARIES   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
.LIB RES

 ************************************************************************
 *****DEBUT****  Poly2 sheet resistance W/O Polycide RES   *****DEBUT****
 ************************************************************************
 .lib 'lt035tsmc.lib' RPO2_PARAM
 ******************* Declaration des distributions **********************
 .lib 'lt035tsmc.lib' RPO2_ABS
 ******************* Model declaration *********************************
 .MODEL  rpo2 R RSH='rpo2_rsh_abs'  DW='rpo2_dw_abs/2' DLR='rpo2_dl_abs/2'
 ******************* Resistor Sub circuit ******************************
 .subckt rpo2 N1 N2 l=1u w=1u
  .param rpo2_dev_rel   = agauss(0,1,r_sigma_rel,r_mult_rel)
  .param rpo2_rsh_rel   = '(1-rpo2_dsp_rsh% *rpo2_dev_rel/2)/\\
```

```
                          (1+rpo2_dsp_rsh% *rpo2_dev_rel/2)'
                          $a cause de l`influence indirecte par SCALE
  .param rpo2_dl_rel    = agauss(0,rpo2_dsp_dl,r_sigma_rel,r_mult_rel)
  .param rpo2_dw_rel    = agauss(0,rpo2_dsp_dw,r_sigma_rel,r_mult_rel)
  .param rpo2_ptcp1_rel = 'rpo2_ptcp1_abs*(1+rpo2_var_rel*rpo2_dev_rel)'
  .param rpo2_ptcp2_rel = 'rpo2_ptcp2_abs*(1+rpo2_var_rel*rpo2_dev_rel)'
  .param rpo2_pvcp1_rel = 'rpo2_pvcp1_abs*(1+rpo2_var_rel*rpo2_dev_rel)'
  .param rpo2_pvcp2_rel = 'rpo2_pvcp2_abs*(1+rpo2_var_rel*rpo2_dev_rel)'

 R1    N1 N2 rpo2 L='l+rpo2_dl_rel' W='w+rpo2_dw_rel'
+SCALE='(1+rpo2_pvcp1_rel*abs(v(N1,N2))\\
         +rpo2_pvcp2_rel*v(N1,N2)*v(N1,N2))\\
       *(1+rpo2_ptcp1_rel*(temper-25.0)\\
         +rpo2_ptcp2_rel*(temper-25.0)*(temper-25.0))\\
         *rpo2_rsh_rel'
 .ends
 ************************************************************************
 *****FIN******  Poly2 sheet resistance W/O Polycide RES  *******FIN*****
 ************************************************************************

.ENDL RES

************************************************************************
************************************************************************
************************************************************************
******                        VARIABLES ABSOLUES                      *******
************************************************************************
************************************************************************
************************************************************************

*********************************MOS_ABS********************************
.LIB MOS_ABS
 .param toxn_dev_abs = agauss(0,1,m_sigma_abs,m_mult_abs) $[-(1),(1)]
 .param toxn    = 'mos_toxn  + mos_var_toxn *toxn_dev_abs'
 .param toxp_dev_abs = agauss(0,1,m_sigma_abs,m_mult_abs) $[-(1),(1)]
 .param toxp    = 'mos_toxp  + mos_var_toxp *toxp_dev_abs'

 .param dvtn_dev_abs = agauss(0,1,m_sigma_abs,m_mult_abs) $[-(1),(1)]
 .param dvtn    = 'mos_dvtn  + mos_var_dvtn *dvtn_dev_abs'
 .param dvtp_dev_abs = agauss(0,1,m_sigma_abs,m_mult_abs) $[-(1),(1)]
 .param dvtp    = 'mos_dvtp  + mos_var_dvtp *dvtp_dev_abs'

 .param cjn     = ' 950.957e-6  +950.957e-6  *dvtn'
 .param cjswn   = ' 284.489e-12 +284.489e-12 *dvtn'
 .param cgon    = '-130.150e-12 + 41.260e-3  *toxn'
 .param cjswgn  = ' 181.651e-12 +181.651e-12 *dvtn'

 .param cjp     = '  1.389e-3   -  1.389e-3  *dvtp'
 .param cjswp   = '386.159e-12  -386.159e-12 *dvtp'
 .param cgop    = '-98.82e-12   + 36.6e-3    *toxp'
 .param cjswgp  = ' 44.154e-12  - 44.154e-12 *dvtp'

 .param dxl_dev_abs = agauss(0,1,m_sigma_abs,m_mult_abs) $[-(1),(1)]
 .param dxl     = 'mos_dxl    + mos_var_dxl  *dxl_dev_abs'
 .param dxw_dev_abs = agauss(0,1,m_sigma_abs,m_mult_abs) $[-(1),(1)]
 .param dxw     = 'mos_dxw    + mos_var_dxw  *dxw_dev_abs'

.ENDL MOS_ABS
*********************************RPO2_ABS********************************
.LIB RPO2_ABS
 .param rpo2_rsh_abs  = gauss(rpo2_rsh  ,rpo2_var_rsh%  ,r_sigma_abs,r_mult_abs)
 .param rpo2_dw_abs   =agauss(rpo2_dw   ,rpo2_var_dw    ,r_sigma_abs,r_mult_abs)
 .param rpo2_dl_abs   =agauss(rpo2_dl   ,rpo2_var_dl    ,r_sigma_abs,r_mult_abs)
 .param rpo2_ptcp1_abs= gauss(rpo2_ptcp1,rpo2_var_ptcp1%,r_sigma_abs,r_mult_abs)
 .param rpo2_ptcp2_abs= gauss(rpo2_ptcp2,rpo2_var_ptcp2%,r_sigma_abs,r_mult_abs)
 .param rpo2_pvcp1_abs= gauss(rpo2_pvcp1,rpo2_var_pvcp1%,r_sigma_abs,r_mult_abs)
 .param rpo2_pvcp2_abs= gauss(rpo2_pvcp2,rpo2_var_pvcp2%,r_sigma_abs,r_mult_abs)
```

```
.ENDL RPO2_ABS
*********************************BJT_ABS*********************************
.LIB BJT_ABS
 .param bjt_dev_abs  = agauss(0,bjt_var_abs,q_sigma_abs,q_mult_abs)
 .param isa  = '1 + bjt_var_is%  *bjt_dev_abs'
 .param bfa  = '1 + bjt_var_bf%  *bjt_dev_abs'
 .param nfa  = '1 - bjt_var_nf%  *bjt_dev_abs'
 .param rba  = '1 - bjt_var_rb%  *bjt_dev_abs'
 .param rea  = '1 - bjt_var_re%  *bjt_dev_abs'
 .param rca  = '1 - bjt_var_rc%  *bjt_dev_abs'
 .param rbma = '1 - bjt_var_rbm% *bjt_dev_abs'
 .param cjea = '1 - bjt_var_cje% *bjt_dev_abs'
 .param cjca = '1 - bjt_var_cjc% *bjt_dev_abs'
.ENDL BJT_ABS

*************************************************************************
*************************************************************************
*************************************************************************
******                    PARAMETRES de CONTROLE                  *******
*************************************************************************
*************************************************************************
*************************************************************************

*********************************MOS_PARAM*********************************
.LIB MOS_PARAM
 .param m_var_abs     = 0.10  $les proportions de var. de base sont
                              $déjà intégré dans les paramètres du BJT
 .param m_var_rel     = 0.10  $signifie 10% variation de base

 .param mos_var_abs   ='m_var_abs/mos_var_base' $[(0.1)/0.1]=[(1)]
 .param mos_var_rel   ='m_var_abs*m_var_rel'  $[(0.1)/0.1*(0.1)]
 .param mos_var_base  ='0.1'                  $[0.1]

 .param m_sigma_abs   = 3
 .param m_sigma_rel   = 3
 .param m_mult_abs    = 1     $0 desactive l'option
 .param m_mult_rel    = 1     $un facteur > 1 cree une bimodale

 .param hdifn = 3.65E-07 hdifp = 3.65E-07

 .param mos_toxn      = 7.5n
 .param mos_toxp      = 7.7n
 .param mos_var_toxn  = 0.5n          $Corner TSMC
 .param mos_var_toxp  = 0.5n          $Corner TSMC

 .param mos_dxl               = 0
 .param mos_var_dxl   = 40n           $Corner TSMC

 .param mos_dxw               = 0
 .param mos_var_dxw   = 60n           $Corner TSMC

 .param mos_dsp_dxn   = 0.0n          $agit sur IDsat et Beta
 .param mos_dsp_dxp   = 0.0n          $

 .param mos_dvtn      = 0
 .param mos_dvtp      = 0
 .param mos_var_dvtn  = 0.1           $Corner TSMC
 .param mos_var_dvtp  = 0.1           $Corner TSMC

 .param mos_dsp_An    = 2.8869        $facteur de mismatch
 .param mos_dsp_Bn    = 0.3465        $facteur de mismatch
 .param mos_dsp_Ap    = 4.9655        $facteur de mismatch
 .param mos_dsp_Bp    = 0.1316        $facteur de mismatch

 .param mos_rcn               = 45          $résistance de contact
 .param mos_rcp               = 90          $résistance de contact
 .param mos_var_rc%   = 1.0           $var. abs.
```

```
 .param mos_dsp_rc%    = 0.1          $var. rel.
.ENDL MOS_PARAM

*********************************BJT_PARAM*********************************
.LIB BJT_PARAM
 .param q_var_abs      = 0.10 $les proportions de var. de base sont
                              $déjà intégré dans les paramètres du BJT
 .param q_var_rel      = 0.10 $signifie 10% variation de base
 .param q_sigma_abs    = 3
 .param q_sigma_rel    = 3
 .param q_mult_abs     = 1
 .param q_mult_rel     = 1

 .param bjt_var_abs    ='q_var_abs/bjt_var_base' $[(0.1)/0.1]=[(1)]
 .param bjt_var_rel    ='q_var_abs*q_var_rel'  $[(0.1)/0.1*(0.1)]
 .param bjt_var_base   ='0.1'                  $[0.1]

 .param bjt_var_is%    = 0.3  $30% de base
 .param bjt_var_bf%    = 0.15 $15% de base
 .param bjt_var_nf%    = 0.015$1.5% de base
                              $facteur dominant de la deviation rel.
 .param bjt_var_rb%    = 0.2  $20% de base
 .param bjt_var_re%    = 0.2  $20% de base
 .param bjt_var_rc%    = 0.2  $20% de base
 .param bjt_var_rbm%   = 0.2  $20% de base
 .param bjt_var_cje%   = 0.05 $5% de base
 .param bjt_var_cjc%   = 0.05 $5% de base

 .param ka_pnp10       = 1    $facteur de mismatch
.ENDL BJT_PARAM

*********************************RPO2_PARAM*********************************
.LIB    RPO2_PARAM
 .param rpo2_var_abs  ='r_var_abs'          $[(0.1)]
 .param rpo2_var_rel  ='r_var_abs*r_var_rel'$[(0.1)*(0.1)]
$$$$$$$$$$$$$$$$$$$
$valeurs nominales$
$$$$$$$$$$$$$$$$$$$
 .param rpo2_dl          = 0.0u
 .param rpo2_dw          = 0.0u
 .param rpo2_rsh   = 50
 .param rpo2_ptcp1 = 1.07e-3
 .param rpo2_ptcp2 = 0
 .param rpo2_pvcp1 =-6.14e-6
 .param rpo2_pvcp2 = 2.33e-4

$$$$$$$$$$$$$$$$$$$$$
$variations absolues$
$$$$$$$$$$$$$$$$$$$$$
 .param rpo2_var_dw   = 0n   $ Choisi par Hugues
 .param rpo2_var_dl   = 0n   $ Choisi par Hugues

 .param rpo2_var_rsh% = 0.01 $'rpo2_var_abs'$ TBD
 .param rpo2_var_ptcp1%      = 0 $'rpo2_var_abs'   $ TBD
 .param rpo2_var_ptcp2%      = 0 $'rpo2_var_abs'   $ TBD
 .param rpo2_var_pvcp1%      = 0 $'rpo2_var_abs'   $ TBD
 .param rpo2_var_pvcp2%      = 0 $'rpo2_var_abs'   $ TBD

$$$$$$$$$$$$$$$$$$$$$$
$dispersions relative$
$$$$$$$$$$$$$$$$$$$$$$
 .param rpo2_dsp_dw    = 0n
 .param rpo2_dsp_dl    = 0n

 .param rpo2_dsp_rsh%  = 0 $'rpo2_var_rel'  $ TBD
 .param rpo2_dsp_ptcp1%= 0 $'rpo2_var_rel'  $ TBD
 .param rpo2_dsp_ptcp2%= 0 $'rpo2_var_rel'  $ TBD
```

```
 .param rpo2_dsp_pvcp1%= 0 $'rpo2_var_rel'   $ TBD
 .param rpo2_dsp_pvcp2%= 0 $'rpo2_var_rel'   $ TBD

.ENDL   RPO2_PARAM
```

Listage 4.4 Bibliothèque tronquée du fichier technologique TSMC 0.35µm adapté à l'analyse Monte Carlo.

```
*
*  ********************************************
*  *             TSMC SPICE MODEL              *
*  ********************************************
*
*    PROCESS :  0.35um MIXED MODE (2P4M,3.3V/5V) POLYCIDE
*    MODEL   :  BSIM3 ( V3.1 )
*    DOC. NO.:  T-035-MM-SP-002
*    VERSION :  2.3
*    DATE    :  Oct 25 , 2001
*    HSPICE VERSION : H98.2 & H98.4
*
*********************************************************************************
*

(...)

*
.LIB MOS
*
 .lib 'lt035tsmc.lib' MOS
*****************************************************************
*                                                               *
*               3.3V NMOS DEVICES MODEL                         *
*                                                               *
*****************************************************************
.MODEL nch.1              NMOS    (                    LMIN    = '1.207E-06-dxl'
+LMAX    = 2.01E-05         WMIN    = '1.183E-06-dxw' WMAX    = '1.9983E-5-dxw'
+LEVEL   = 49               TNOM    = 25              VERSION = 3.1

(...)

+CALCACM = 1               SFVTFLAG = 0              VFBFLAG  = 1          )
*
.ENDL MOS
*********************************************************************************
*
.LIB BIP
*
 .lib 'lt035tsmc.lib' BJT
*****************************************************************
*          MODEL OF P+/NW/PSUB PNP 10X10 VERTICAL BIPOLAR       *
*****************************************************************
.MODEL pnp10 PNP (                                  LEVEL  = 1
+ BF      = 6.35             NF      = 1.01           ISE    = 2.95E-17
+ NE      = 1.9              IS      = 2.95E-17       RB     = 71
+ IRB     = 9.5E-4           RBM     = 0.1            RE     = 2.35
+ IKF     = 1.95E-3          NKF     = 0.549          VAF    = 300
+ BR      = 0.01116          NR      = 1              ISC    = 2.95E-17
+ NC      = 1.1              RC      = 21.08          IKR    = 0.087818
+ VAR     = 7.8              TBF1    = 6.7E-3         TBF2   = 8E-6
+ TNE1    = 9E-5             TNF1    = 9.5E-5         TRB1   = 3E-5
+ TIRB1   = 9E-4             TRM1    = 9.44068E-6     TRE1   = 5E-4
+ TIKF1   = -3.5E-3          TVAF1   = -9E-4          TBR1   = -7E-4
+ TBR2    = 9.5E-6           TNR1    = 9E-5           TNC1   = 1.50731E-3
+ XTB     = 0                XTI     = 3              TRC1   = 0
+ TIKR1   = -3.9E-3          EG      = 1.18           CJE    = 1.55101E-13
+ VJE     = 1.23623          MJE     = 0.662557       FC     = 0.5
+ CJC     = 3.131405E-14     VJC     = 0.552774       MJC    = 0.32993
+ TLEVC   = 1                CTE     = 4.87874E-4     CTC    = 2.63294E-3
```

```
+ TVJE    = 2.16319E-3       TVJC    = 2.955E-3         TREF    = 25
+ SUBS    = 1                TLEV    = 0                )
*
.ENDL BIP
********************************************************************************
*
.lib RES
********************************************************************************
.subckt rpo2 n1 n2 l=length w=width
.param rsh=50 dw=0u ptc1=1.07e-3 ptc2=0 pvc1=-6.14e-6 pvc2=2.33e-4 pt='temper'
.param tfac='1.0+ptc1*(pt-25.0)+ptc2*(pt-25.0)*(pt-25.0)'
r1 n1 n2 'rsh*l/(w-dw)*(1+pvc1*abs(v(n2,n1))+pvc2*v(n2,n1)*v(n2,n1))*tfac'
.ends rpo2
*
.endl RES
********************************************************************************
```

Annexe III. Résultats d'optimisations : Statistiques de convergence

Les statistiques de convergence présentent la progression de la convergence de la population utilisée par l'AG. On relate, au fil des générations qui se succèdent, le pointage associé *au meilleur individu* ainsi que celui associé *au plus faible*. La moyenne des pointages associés à cette population est aussi reportée.

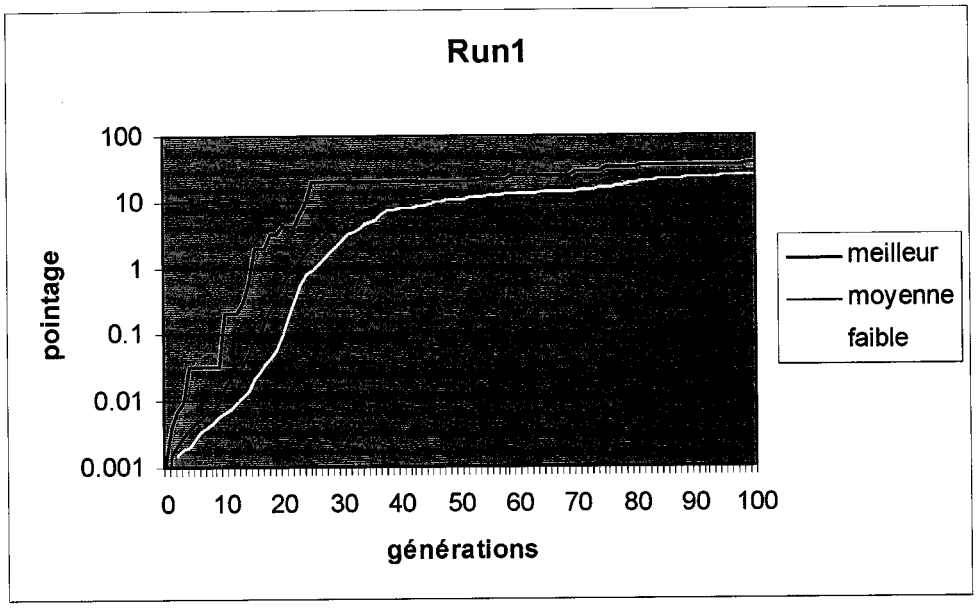


Figure 4.1 Statistique de convergence pour l'essai 1.

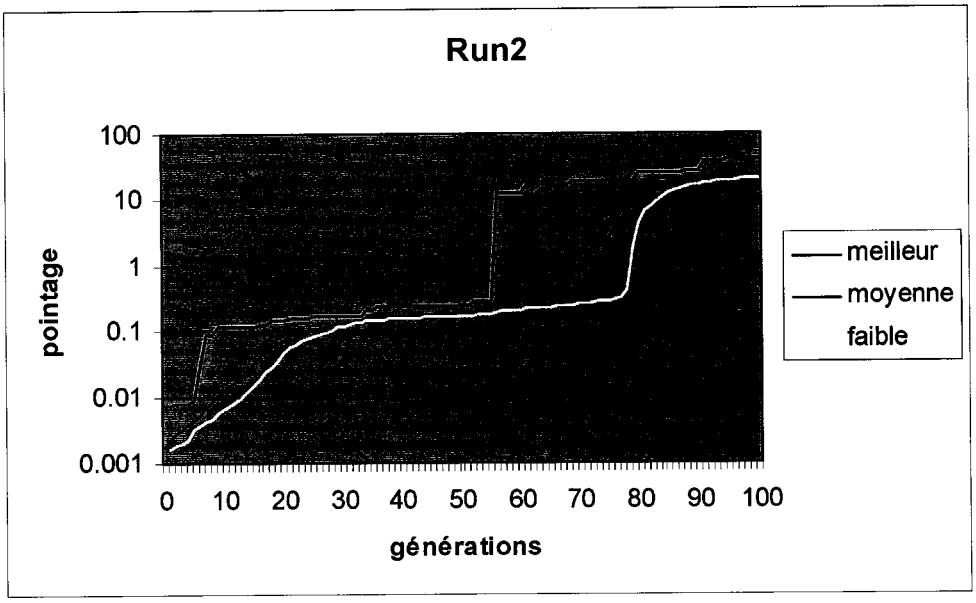


Figure 4.2 Statistique de convergence pour l'essai 2.

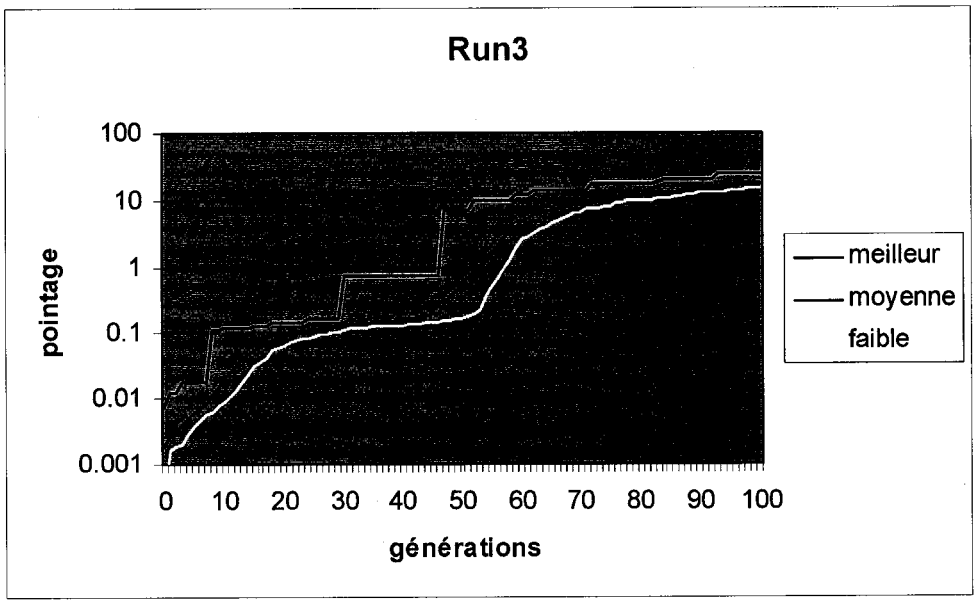


Figure 4.3 Statistique de convergence pour l'essai 3.

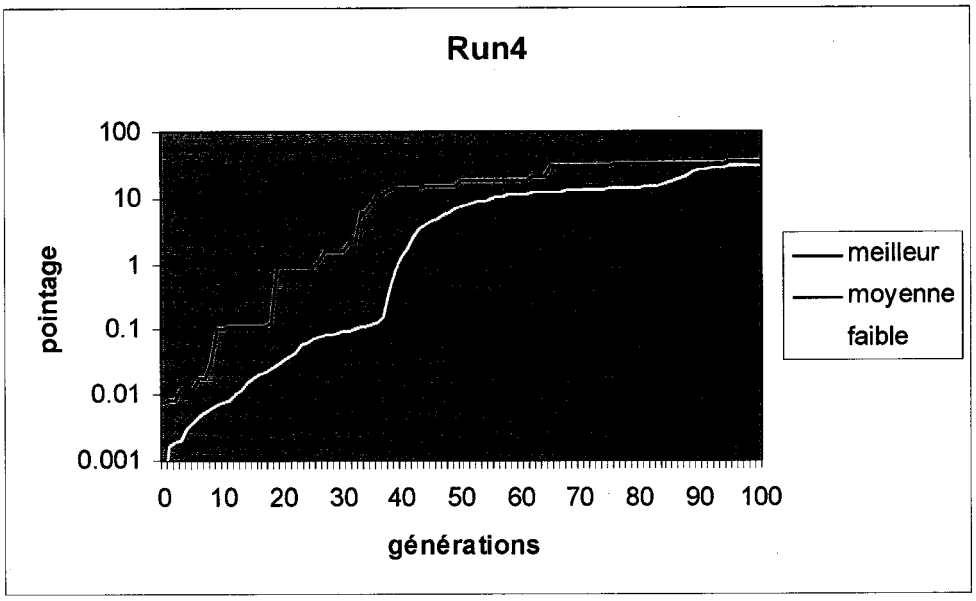


Figure 4.4 Statistique de convergence pour l'essai 4.

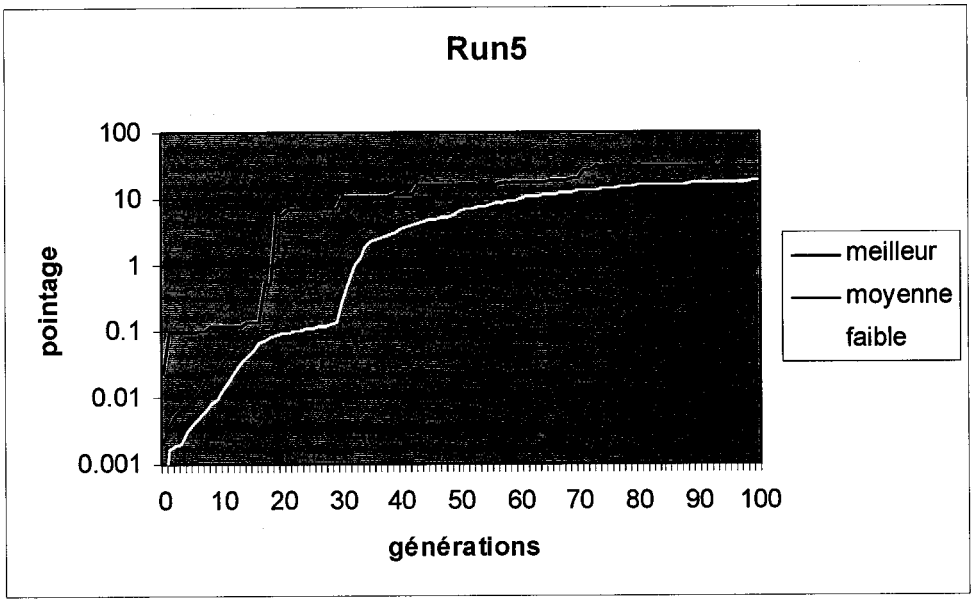


Figure 4.5 Statistique de convergence pour l'essai 5.

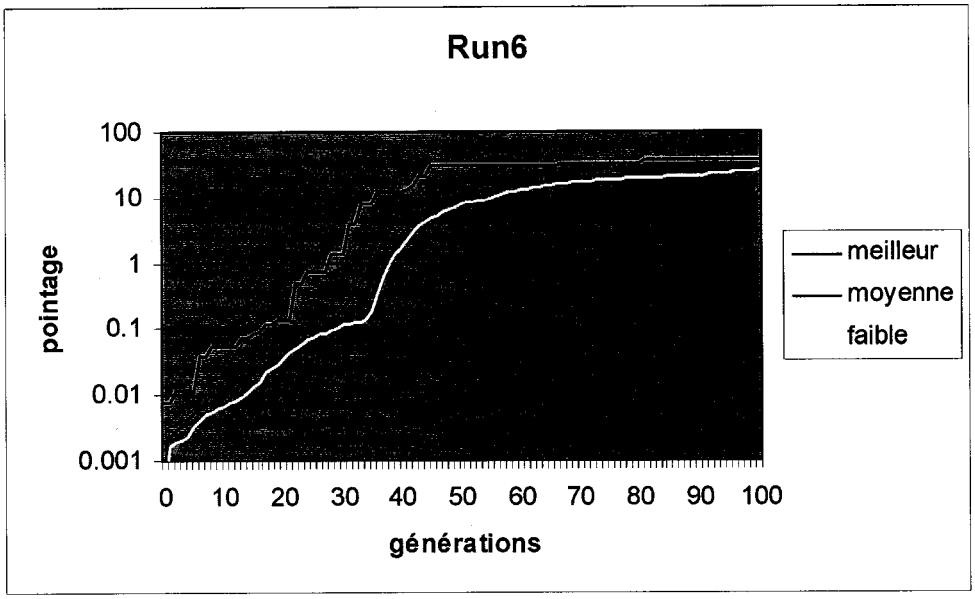


Figure 4.6 Statistique de convergence pour l'essai 6.

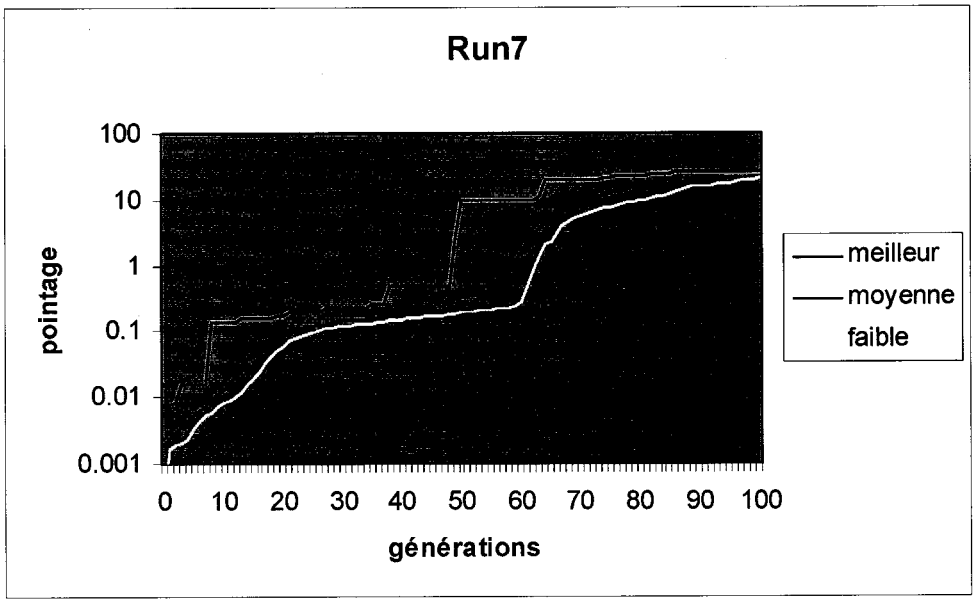


Figure 4.7 Statistique de convergence pour l'essai 7.

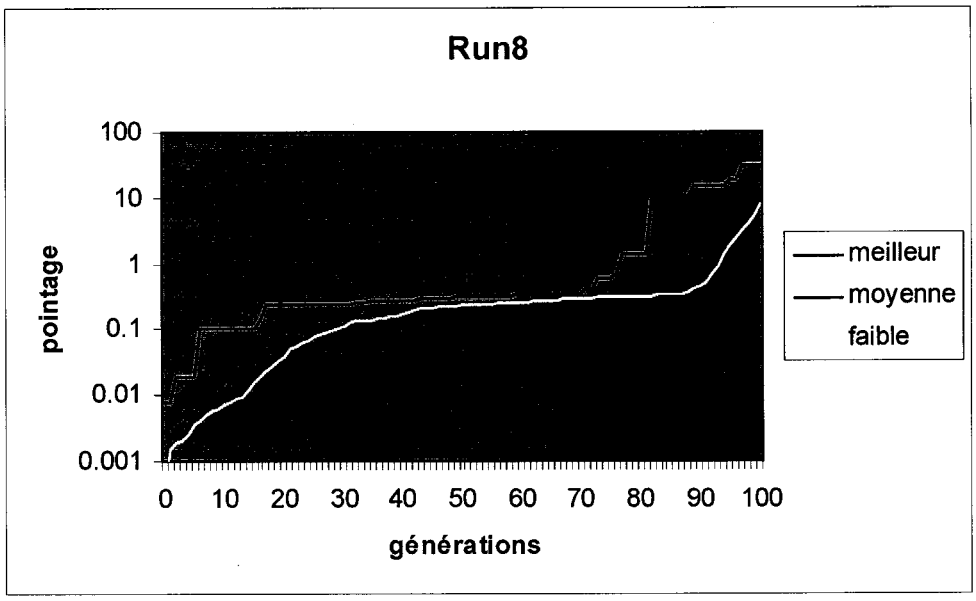


Figure 4.8 Statistique de convergence pour l'essai 8.

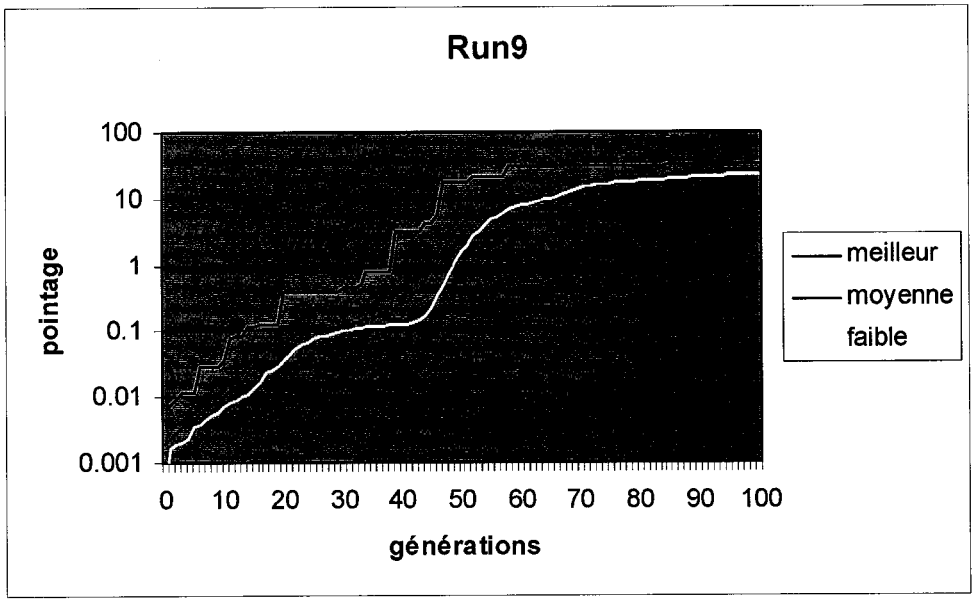


Figure 4.9 Statistique de convergence pour l'essai 9.

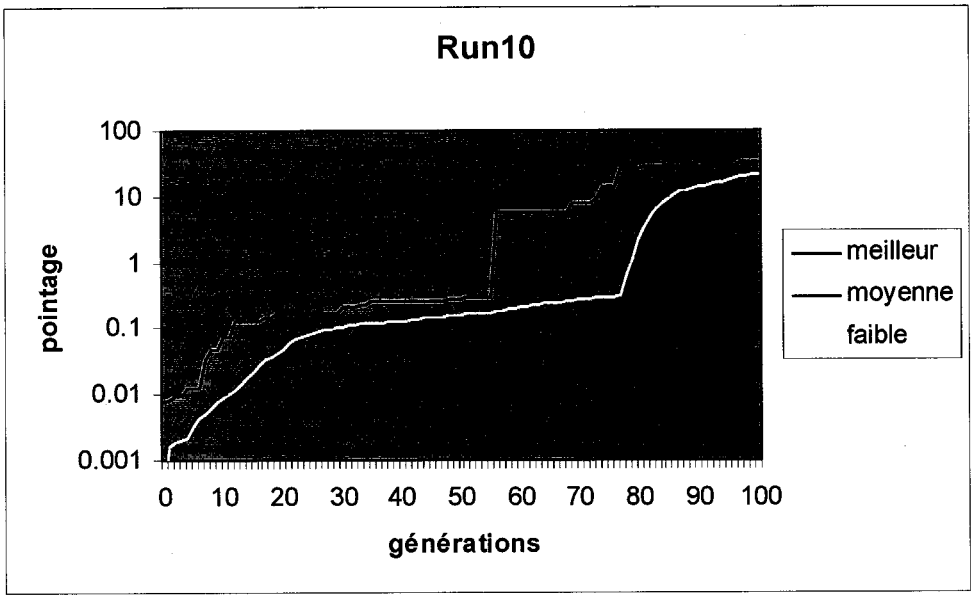


Figure 4.10 Statistique de convergence pour l'essai 10.

Annexe IV. Résultats d'optimisations : Paramètres optimisés pour le circuit

Cette section présente les paramètres finaux trouvés lors des 10 essais distincts d'optimisation sur le circuit de référence de tension Chang. On relate uniquement les valeurs finales découvertes étant donné que la description du circuit reste toujours celle présentée à l'Annexe I.

Listage 4.5 Paramètres optimaux de l'essai #1 (Run1)

```
.param l1=opt(1.5u,1u,5u,500n)
.param l2=l1
.param l3=l1
.param l4=opt(2.5u,1u,5u,500n)
.param l5=l4
.param l6=opt(2u,1u,5u,500n)
.param l7=l6
.param l8=opt(1u,1u,5u,500n)
.param l9=l8
.param l10=opt(5u,1u,5u,500n)
.param l11=opt(1.5u,1u,5u,500n)
.param l12=l11
.param l13=l4
.param l14=l8
.param l15=opt(4.5u,1u,5u,500n)
.param l16=opt(6u,1u,20u,1u)
.param l17=opt(14u,1u,20u,1u)
.param w=opt(5u,5u,20u,5u)
.param m1=opt(7,1,10,1)
.param m2=opt(7,1,10,1)
.param m3=opt(7,1,10,1)
.param m4=opt(1,1,10,1)
.param m5=opt(9,1,10,1)
.param m6=opt(4,1,10,1)
.param m7=m6
.param m8=opt(5,1,10,1)
.param m9=m8
.param m10=opt(7,1,10,1)
.param m11=opt(1,1,10,1)
.param m12=opt(10,1,10,1)
.param m13=opt(8,1,10,1)
.param m14=opt(7,1,10,1)
.param m15=opt(6,1,10,1)
.param m16=opt(5,1,10,1)
.param m17=opt(9,1,10,1)
.param c1=opt(5.6p,0,10p,100f)
.param c2=opt(0,0,10p,100f)
.param c3=opt(700f,0,10p,100f)
.param c4=opt(8p,0,50p,1p)
.param lr1=opt(1.4m,0,2m,50u)
.param lr2=opt(2.85m,0,4m,50u)
.param lr3=opt(150u,0,2m,50u)
```

```
.param wr1=opt(1u,1u,5u,1u)
.param wr2=wr1
.param wr3=wr1
.param mq1=opt(1,1,1,1)
.param mq2=opt(1,1,1,1)
.param mq3=opt(6,1,10,1)
.param mq4=opt(1,1,1,1)
```

Listage 4.6 Paramètres optimaux de l'essai #2 (Run2)

```
.param l1=opt(5u,1u,5u,500n)
.param l2=l1
.param l3=l1
.param l4=opt(2u,1u,5u,500n)
.param l5=l4
.param l6=opt(3u,1u,5u,500n)
.param l7=l6
.param l8=opt(1.5u,1u,5u,500n)
.param l9=l8
.param l10=opt(4u,1u,5u,500n)
.param l11=opt(1.5u,1u,5u,500n)
.param l12=l11
.param l13=l4
.param l14=l8
.param l15=opt(5u,1u,5u,500n)
.param l16=opt(19u,1u,20u,1u)
.param l17=opt(9u,1u,20u,1u)
.param w=opt(5u,5u,20u,5u)
.param m1=opt(9,1,10,1)
.param m2=opt(9,1,10,1)
.param m3=opt(8,1,10,1)
.param m4=opt(4,1,10,1)
.param m5=opt(4,1,10,1)
.param m6=opt(2,1,10,1)
.param m7=m6
.param m8=opt(4,1,10,1)
.param m9=m8
.param m10=opt(9,1,10,1)
.param m11=opt(1,1,10,1)
.param m12=opt(9,1,10,1)
.param m13=opt(6,1,10,1)
.param m14=opt(10,1,10,1)
.param m15=opt(8,1,10,1)
.param m16=opt(3,1,10,1)
.param m17=opt(8,1,10,1)
.param c1=opt(6.2p,0,10p,100f)
.param c2=opt(0,0,10p,100f)
.param c3=opt(1p,0,10p,100f)
.param c4=opt(37p,0,50p,1p)
.param lr1=opt(250u,0,2m,50u)
.param lr2=opt(3.8m,0,4m,50u)
.param lr3=opt(50u,0,2m,50u)
.param wr1=opt(1u,1u,5u,1u)
.param wr2=wr1
.param wr3=wr1
.param mq1=opt(1,1,1,1)
.param mq2=opt(1,1,1,1)
.param mq3=opt(6,1,10,1)
.param mq4=opt(1,1,1,1)
```

Listage 4.7 Paramètres optimaux de l'essai #3 (Run3)

```
.param l1=opt(1.5u,1u,5u,500n)
.param l2=l1
.param l3=l1
.param l4=opt(1.5u,1u,5u,500n)
.param l5=l4
.param l6=opt(5u,1u,5u,500n)
.param l7=l6
.param l8=opt(1u,1u,5u,500n)
.param l9=l8
.param l10=opt(5u,1u,5u,500n)
.param l11=opt(3u,1u,5u,500n)
.param l12=l11
.param l13=l4
.param l14=l8
.param l15=opt(1u,1u,5u,500n)
.param l16=opt(11u,1u,20u,1u)
.param l17=opt(6u,1u,20u,1u)
.param w=opt(15u,5u,20u,5u)
.param m1=opt(10,1,10,1)
.param m2=opt(10,1,10,1)
.param m3=opt(9,1,10,1)
.param m4=opt(2,1,10,1)
.param m5=opt(5,1,10,1)
.param m6=opt(7,1,10,1)
.param m7=m6
.param m8=opt(4,1,10,1)
.param m9=m8
.param m10=opt(7,1,10,1)
.param m11=opt(1,1,10,1)
.param m12=opt(9,1,10,1)
.param m13=opt(10,1,10,1)
.param m14=opt(10,1,10,1)
.param m15=opt(2,1,10,1)
.param m16=opt(6,1,10,1)
.param m17=opt(1,1,10,1)
.param c1=opt(0.0032n,0,10p,100f)
.param c2=opt(0,0,10p,100f)
.param c3=opt(700f,0,10p,100f)
.param c4=opt(19p,0,50p,1p)
.param lr1=opt(650u,0,2m,50u)
.param lr2=opt(3.95m,0,4m,50u)
.param lr3=opt(150u,0,2m,50u)
.param wr1=opt(1u,1u,5u,1u)
.param wr2=wr1
.param wr3=wr1
.param mq1=opt(1,1,1,1)
.param mq2=opt(1,1,1,1)
.param mq3=opt(7,1,10,1)
.param mq4=opt(1,1,1,1)
```

Listage 4.8 Paramètres optimaux de l'essai #4 (Run4)

```
.param l1=opt(4u,1u,5u,500n)
.param l2=l1
.param l3=l1
.param l4=opt(3.5u,1u,5u,500n)
.param l5=l4
.param l6=opt(4.5u,1u,5u,500n)
.param l7=l6
.param l8=opt(1.5u,1u,5u,500n)
.param l9=l8
.param l10=opt(2.5u,1u,5u,500n)
.param l11=opt(3.5u,1u,5u,500n)
.param l12=l11
.param l13=l4
.param l14=l8
.param l15=opt(3.5u,1u,5u,500n)
.param l16=opt(13u,1u,20u,1u)
.param l17=opt(11u,1u,20u,1u)
.param w=opt(10u,5u,20u,5u)
.param m1=opt(10,1,10,1)
.param m2=opt(10,1,10,1)
.param m3=opt(7,1,10,1)
.param m4=opt(2,1,10,1)
.param m5=opt(10,1,10,1)
.param m6=opt(4,1,10,1)
.param m7=m6
.param m8=opt(1,1,10,1)
.param m9=m8
.param m10=opt(10,1,10,1)
.param m11=opt(2,1,10,1)
.param m12=opt(8,1,10,1)
.param m13=opt(8,1,10,1)
.param m14=opt(4,1,10,1)
.param m15=opt(1,1,10,1)
.param m16=opt(5,1,10,1)
.param m17=opt(3,1,10,1)
.param c1=opt(600f,0,10p,100f)
.param c2=opt(0,0,10p,100f)
.param c3=opt(0,0,10p,100f)
.param c4=opt(16p,0,50p,1p)
.param lr1=opt(900u,0,2m,50u)
.param lr2=opt(4m,0,4m,50u)
.param lr3=opt(50u,0,2m,50u)
.param wr1=opt(1u,1u,5u,1u)
.param wr2=wr1
.param wr3=wr1
.param mq1=opt(1,1,1,1)
.param mq2=opt(1,1,1,1)
.param mq3=opt(6,1,10,1)
.param mq4=opt(1,1,1,1)
```

Listage 4.9 Paramètres optimaux de l'essai #5 (Run5)

```
.param l1=opt(3.5u,1u,5u,500n)
.param l2=l1
.param l3=l1
.param l4=opt(3u,1u,5u,500n)
.param l5=l4
.param l6=opt(4.5u,1u,5u,500n)
.param l7=l6
.param l8=opt(1.5u,1u,5u,500n)
.param l9=l8
.param l10=opt(5u,1u,5u,500n)
.param l11=opt(1.5u,1u,5u,500n)
.param l12=l11
.param l13=l4
.param l14=l8
.param l15=opt(1.5u,1u,5u,500n)
.param l16=opt(15u,1u,20u,1u)
.param l17=opt(17u,1u,20u,1u)
.param w=opt(10u,5u,20u,5u)
.param m1=opt(9,1,10,1)
.param m2=opt(9,1,10,1)
.param m3=opt(7,1,10,1)
.param m4=opt(2,1,10,1)
.param m5=opt(10,1,10,1)
.param m6=opt(5,1,10,1)
.param m7=m6
.param m8=opt(4,1,10,1)
.param m9=m8
.param m10=opt(7,1,10,1)
.param m11=opt(1,1,10,1)
.param m12=opt(10,1,10,1)
.param m13=opt(9,1,10,1)
.param m14=opt(6,1,10,1)
.param m15=opt(8,1,10,1)
.param m16=opt(7,1,10,1)
.param m17=opt(1,1,10,1)
.param c1=opt(5.6p,0,10p,100f)
.param c2=opt(0,0,10p,100f)
.param c3=opt(1.2p,0,10p,100f)
.param c4=opt(31p,0,50p,1p)
.param lr1=opt(1.5m,0,2m,50u)
.param lr2=opt(3.65m,0,4m,50u)
.param lr3=opt(250u,0,2m,50u)
.param wr1=opt(3u,1u,5u,1u)
.param wr2=wr1
.param wr3=wr1
.param mq1=opt(1,1,1,1)
.param mq2=opt(1,1,1,1)
.param mq3=opt(7,1,10,1)
.param mq4=opt(1,1,1,1)
```

Listage 4.10 Paramètres optimaux de l'essai #6 (Run6)

```
.param l1=opt(2u,1u,5u,500n)
.param l2=l1
.param l3=l1
.param l4=opt(4.5u,1u,5u,500n)
.param l5=l4
.param l6=opt(5u,1u,5u,500n)
.param l7=l6
.param l8=opt(1u,1u,5u,500n)
.param l9=l8
.param l10=opt(4.5u,1u,5u,500n)
.param l11=opt(1.5u,1u,5u,500n)
.param l12=l11
.param l13=l4
.param l14=l8
.param l15=opt(2u,1u,5u,500n)
.param l16=opt(16u,1u,20u,1u)
.param l17=opt(7u,1u,20u,1u)
.param w=opt(15u,5u,20u,5u)
.param m1=opt(6,1,10,1)
.param m2=opt(6,1,10,1)
.param m3=opt(10,1,10,1)
.param m4=opt(4,1,10,1)
.param m5=opt(6,1,10,1)
.param m6=opt(9,1,10,1)
.param m7=m6
.param m8=opt(4,1,10,1)
.param m9=m8
.param m10=opt(10,1,10,1)
.param m11=opt(1,1,10,1)
.param m12=opt(9,1,10,1)
.param m13=opt(9,1,10,1)
.param m14=opt(9,1,10,1)
.param m15=opt(5,1,10,1)
.param m16=opt(7,1,10,1)
.param m17=opt(9,1,10,1)
.param c1=opt(200f,0,10p,100f)
.param c2=opt(0,0,10p,100f)
.param c3=opt(800f,0,10p,100f)
.param c4=opt(31p,0,50p,1p)
.param lr1=opt(1.15m,0,2m,50u)
.param lr2=opt(4m,0,4m,50u)
.param lr3=opt(50u,0,2m,50u)
.param wr1=opt(1u,1u,5u,1u)
.param wr2=wr1
.param wr3=wr1
.param mq1=opt(1,1,1,1)
.param mq2=opt(1,1,1,1)
.param mq3=opt(10,1,10,1)
.param mq4=opt(1,1,1,1)
```

Listage 4.11 Paramètres optimaux de l'essai #7 (Run7)

```
.param l1=opt(2.5u,1u,5u,500n)
.param l2=l1
.param l3=l1
.param l4=opt(1u,1u,5u,500n)
.param l5=l4
.param l6=opt(4u,1u,5u,500n)
.param l7=l6
.param l8=opt(2u,1u,5u,500n)
.param l9=l8
.param l10=opt(5u,1u,5u,500n)
.param l11=opt(2u,1u,5u,500n)
.param l12=l11
.param l13=l4
.param l14=l8
.param l15=opt(5u,1u,5u,500n)
.param l16=opt(19u,1u,20u,1u)
.param l17=opt(3u,1u,20u,1u)
.param w=opt(15u,5u,20u,5u)
.param m1=opt(8,1,10,1)
.param m2=opt(8,1,10,1)
.param m3=opt(10,1,10,1)
.param m4=opt(5,1,10,1)
.param m5=opt(10,1,10,1)
.param m6=opt(10,1,10,1)
.param m7=m6
.param m8=opt(4,1,10,1)
.param m9=m8
.param m10=opt(8,1,10,1)
.param m11=opt(1,1,10,1)
.param m12=opt(10,1,10,1)
.param m13=opt(10,1,10,1)
.param m14=opt(4,1,10,1)
.param m15=opt(4,1,10,1)
.param m16=opt(10,1,10,1)
.param m17=opt(8,1,10,1)
.param c1=opt(5.4p,0,10p,100f)
.param c2=opt(0,0,10p,100f)
.param c3=opt(1p,0,10p,100f)
.param c4=opt(0,0,50p,1p)
.param lr1=opt(500u,0,2m,50u)
.param lr2=opt(3.6m,0,4m,50u)
.param lr3=opt(50u,0,2m,50u)
.param wr1=opt(1u,1u,5u,1u)
.param wr2=wr1
.param wr3=wr1
.param mq1=opt(1,1,1,1)
.param mq2=opt(1,1,1,1)
.param mq3=opt(5,1,10,1)
.param mq4=opt(1,1,1,1)
```

Listage 4.12 Paramètres optimaux de l'essai #8 (Run8)

```
.param l1=opt(5u,1u,5u,500n)
.param l2=l1
.param l3=l1
.param l4=opt(3.5u,1u,5u,500n)
.param l5=l4
.param l6=opt(4u,1u,5u,500n)
.param l7=l6
.param l8=opt(3u,1u,5u,500n)
.param l9=l8
.param l10=opt(4u,1u,5u,500n)
.param l11=opt(4.5u,1u,5u,500n)
.param l12=l11
.param l13=l4
.param l14=l8
.param l15=opt(1u,1u,5u,500n)
.param l16=opt(16u,1u,20u,1u)
.param l17=opt(7u,1u,20u,1u)
.param w=opt(20u,5u,20u,5u)
.param m1=opt(10,1,10,1)
.param m2=opt(10,1,10,1)
.param m3=opt(7,1,10,1)
.param m4=opt(5,1,10,1)
.param m5=opt(10,1,10,1)
.param m6=opt(1,1,10,1)
.param m7=m6
.param m8=opt(2,1,10,1)
.param m9=m8
.param m10=opt(10,1,10,1)
.param m11=opt(1,1,10,1)
.param m12=opt(10,1,10,1)
.param m13=opt(8,1,10,1)
.param m14=opt(7,1,10,1)
.param m15=opt(2,1,10,1)
.param m16=opt(2,1,10,1)
.param m17=opt(5,1,10,1)
.param c1=opt(400f,0,10p,100f)
.param c2=opt(0,0,10p,100f)
.param c3=opt(200f,0,10p,100f)
.param c4=opt(49p,0,50p,1p)
.param lr1=opt(350u,0,2m,50u)
.param lr2=opt(3.95m,0,4m,50u)
.param lr3=opt(150u,0,2m,50u)
.param wr1=opt(1u,1u,5u,1u)
.param wr2=wr1
.param wr3=wr1
.param mq1=opt(1,1,1,1)
.param mq2=opt(1,1,1,1)
.param mq3=opt(5,1,10,1)
.param mq4=opt(1,1,1,1)
```

Listage 4.13 Paramètres optimaux de l'essai #9 (Run9)

```
.param l1=opt(2u,1u,5u,500n)
.param l2=l1
.param l3=l1
.param l4=opt(4.5u,1u,5u,500n)
.param l5=l4
.param l6=opt(3u,1u,5u,500n)
.param l7=l6
.param l8=opt(2u,1u,5u,500n)
.param l9=l8
.param l10=opt(5u,1u,5u,500n)
.param l11=opt(2.5u,1u,5u,500n)
.param l12=l11
.param l13=l4
.param l14=l8
.param l15=opt(2.5u,1u,5u,500n)
.param l16=opt(12u,1u,20u,1u)
.param l17=opt(10u,1u,20u,1u)
.param w=opt(10u,5u,20u,5u)
.param m1=opt(7,1,10,1)
.param m2=opt(7,1,10,1)
.param m3=opt(7,1,10,1)
.param m4=opt(3,1,10,1)
.param m5=opt(10,1,10,1)
.param m6=opt(3,1,10,1)
.param m7=m6
.param m8=opt(4,1,10,1)
.param m9=m8
.param m10=opt(10,1,10,1)
.param m11=opt(1,1,10,1)
.param m12=opt(9,1,10,1)
.param m13=opt(7,1,10,1)
.param m14=opt(9,1,10,1)
.param m15=opt(9,1,10,1)
.param m16=opt(1,1,10,1)
.param m17=opt(7,1,10,1)
.param c1=opt(300f,0,10p,100f)
.param c2=opt(0,0,10p,100f)
.param c3=opt(700f,0,10p,100f)
.param c4=opt(46p,0,50p,1p)
.param lr1=opt(900u,0,2m,50u)
.param lr2=opt(3.05m,0,4m,50u)
.param lr3=opt(50u,0,2m,50u)
.param wr1=opt(1u,1u,5u,1u)
.param wr2=wr1
.param wr3=wr1
.param mq1=opt(1,1,1,1)
.param mq2=opt(1,1,1,1)
.param mq3=opt(8,1,10,1)
.param mq4=opt(1,1,1,1)
```

Listage 4.14 Paramètres optimaux de l'essai #10 (Run10)

```
.param l1=opt(5u,1u,5u,500n)
.param l2=l1
.param l3=l1
.param l4=opt(3.5u,1u,5u,500n)
.param l5=l4
.param l6=opt(4u,1u,5u,500n)
.param l7=l6
.param l8=opt(2u,1u,5u,500n)
.param l9=l8
.param l10=opt(4u,1u,5u,500n)
.param l11=opt(2u,1u,5u,500n)
.param l12=l11
.param l13=l4
.param l14=l8
.param l15=opt(5u,1u,5u,500n)
.param l16=opt(19u,1u,20u,1u)
.param l17=opt(19u,1u,20u,1u)
.param w=opt(20u,5u,20u,5u)
.param m1=opt(7,1,10,1)
.param m2=opt(7,1,10,1)
.param m3=opt(9,1,10,1)
.param m4=opt(7,1,10,1)
.param m5=opt(8,1,10,1)
.param m6=opt(2,1,10,1)
.param m7=m6
.param m8=opt(2,1,10,1)
.param m9=m8
.param m10=opt(10,1,10,1)
.param m11=opt(1,1,10,1)
.param m12=opt(9,1,10,1)
.param m13=opt(6,1,10,1)
.param m14=opt(4,1,10,1)
.param m15=opt(8,1,10,1)
.param m16=opt(2,1,10,1)
.param m17=opt(10,1,10,1)
.param c1=opt(200f,0,10p,100f)
.param c2=opt(0,0,10p,100f)
.param c3=opt(1p,0,10p,100f)
.param c4=opt(46p,0,50p,1p)
.param lr1=opt(500u,0,2m,50u)
.param lr2=opt(2.95m,0,4m,50u)
.param lr3=opt(100u,0,2m,50u)
.param wr1=opt(1u,1u,5u,1u)
.param wr2=wr1
.param wr3=wr1
.param mq1=opt(1,1,1,1)
.param mq2=opt(1,1,1,1)
.param mq3=opt(10,1,10,1)
.param mq4=opt(1,1,1,1)
```

Annexe V. Résultats d'optimisations : Courbes de simulation

Cette section présente les courbes de simulation complètes et finales obtenues pour chacune des performances observées au terme des 10 essais distincts d'optimisation. Les performances de chaque circuit tiennent sur 2 feuilles et elle sont marquées par un indice de l'essai (Run#) dans le titre. Le détail de la description de ces courbes est présenté au Tableau 4.1.

Tableau 4.1 Détails supplémentaires sur les courbes de performance

Performance	Description	Premier Balayage	Second Balayage
BOOT	Démarrage autonome	Temps	Monte Carlo (30 essais)
PSRR	Immunité aux variations de l'alimentation	Fréquence	Température (-40 à 85°C)
TEMPCO	Stabilité de la tension de sortie	Température	Alimentation (±10%), Monte Carlo (30 essais)
COURANT	Consommation en courant	Température	Monte Carlo (30 essais)
NOIBF NOIMF	Bruit intrinsèque RMS	Température	-
AREA	Surface occupée	-	-

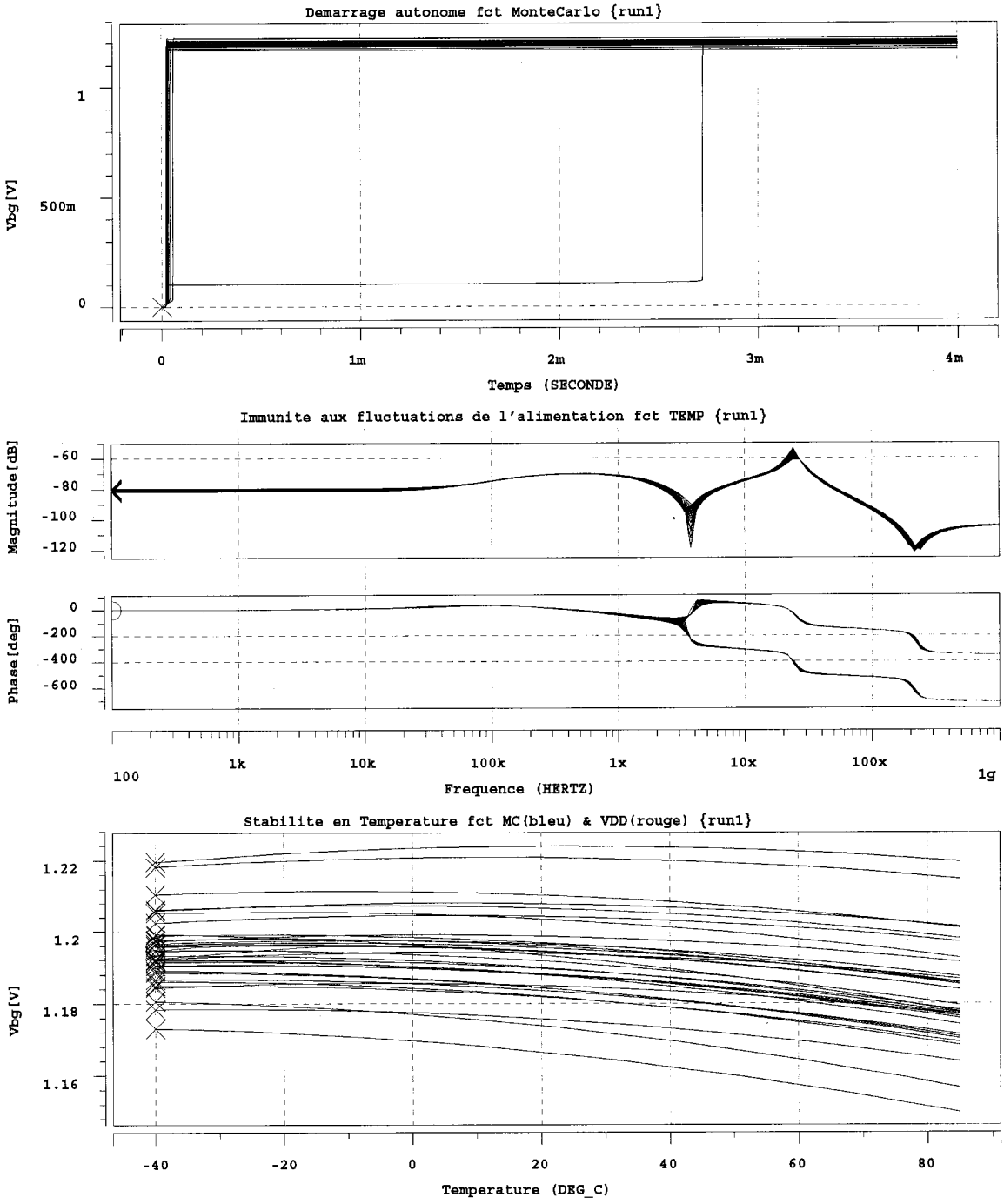


Figure 4.11 Courbes des performances de l'essai 1: (a) démarrage autonome, (b) immunité aux fluctuations de l'alimentation, (c) stabilité en température.

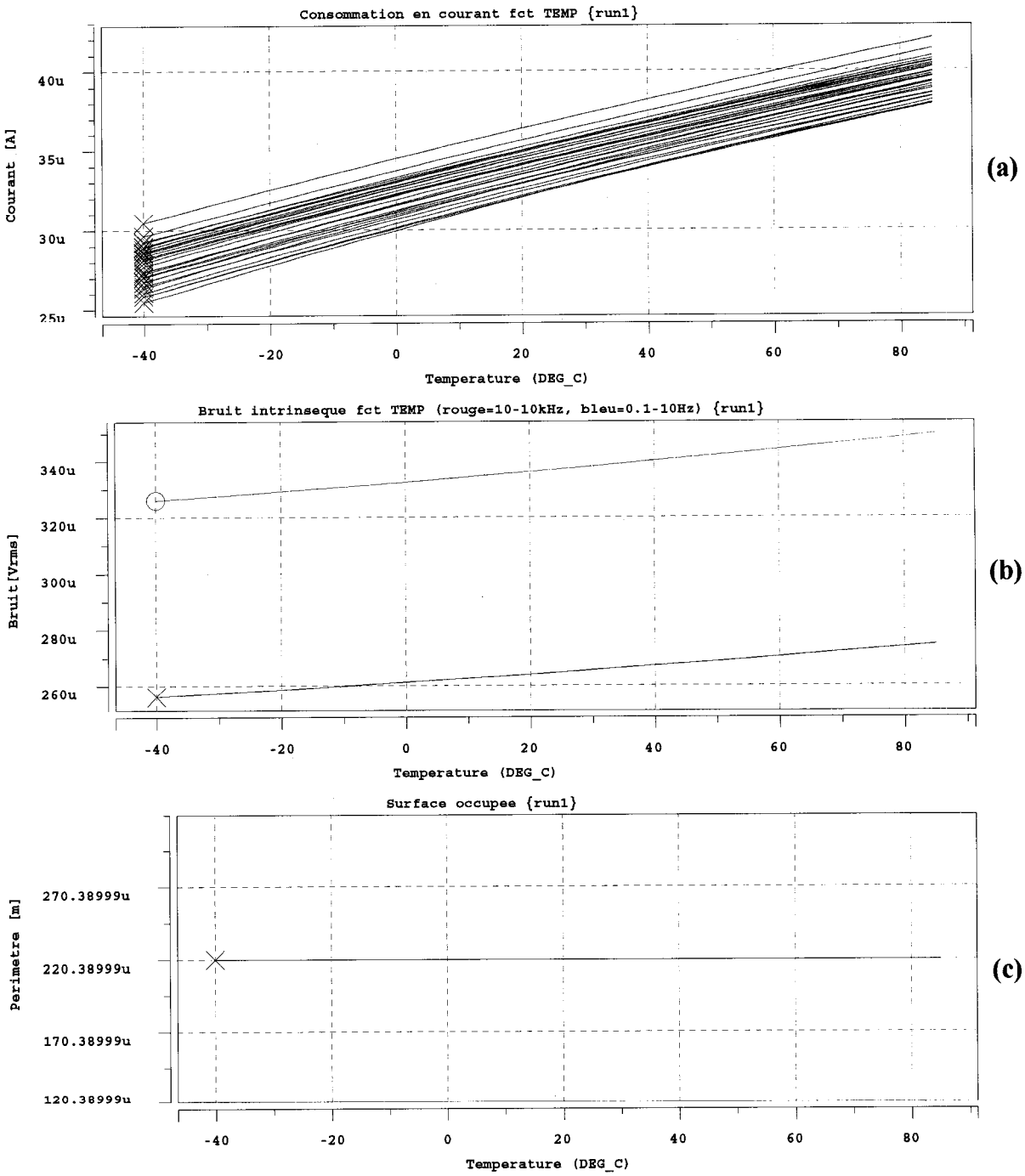


Figure 4.12 Courbes de performances de l'essai 1 : (a) consommation en courant, (b) bruit intrinsèque, (c) surface occupée.

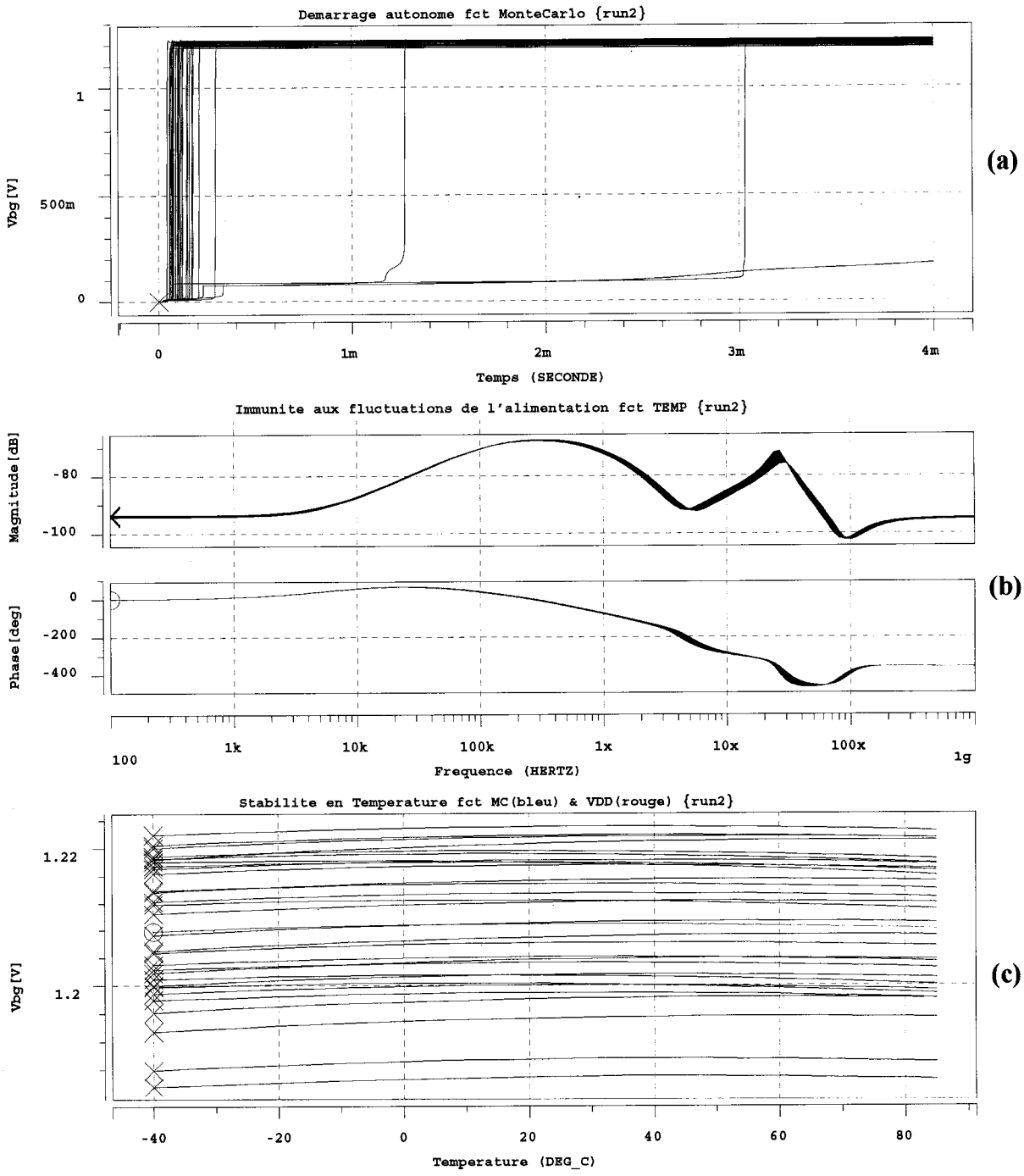


Figure 4.13 Courbes des performances de l'essai 2: (a) démarrage autonome, (b) immunité aux fluctuations de l'alimentation, (c) stabilité en température.

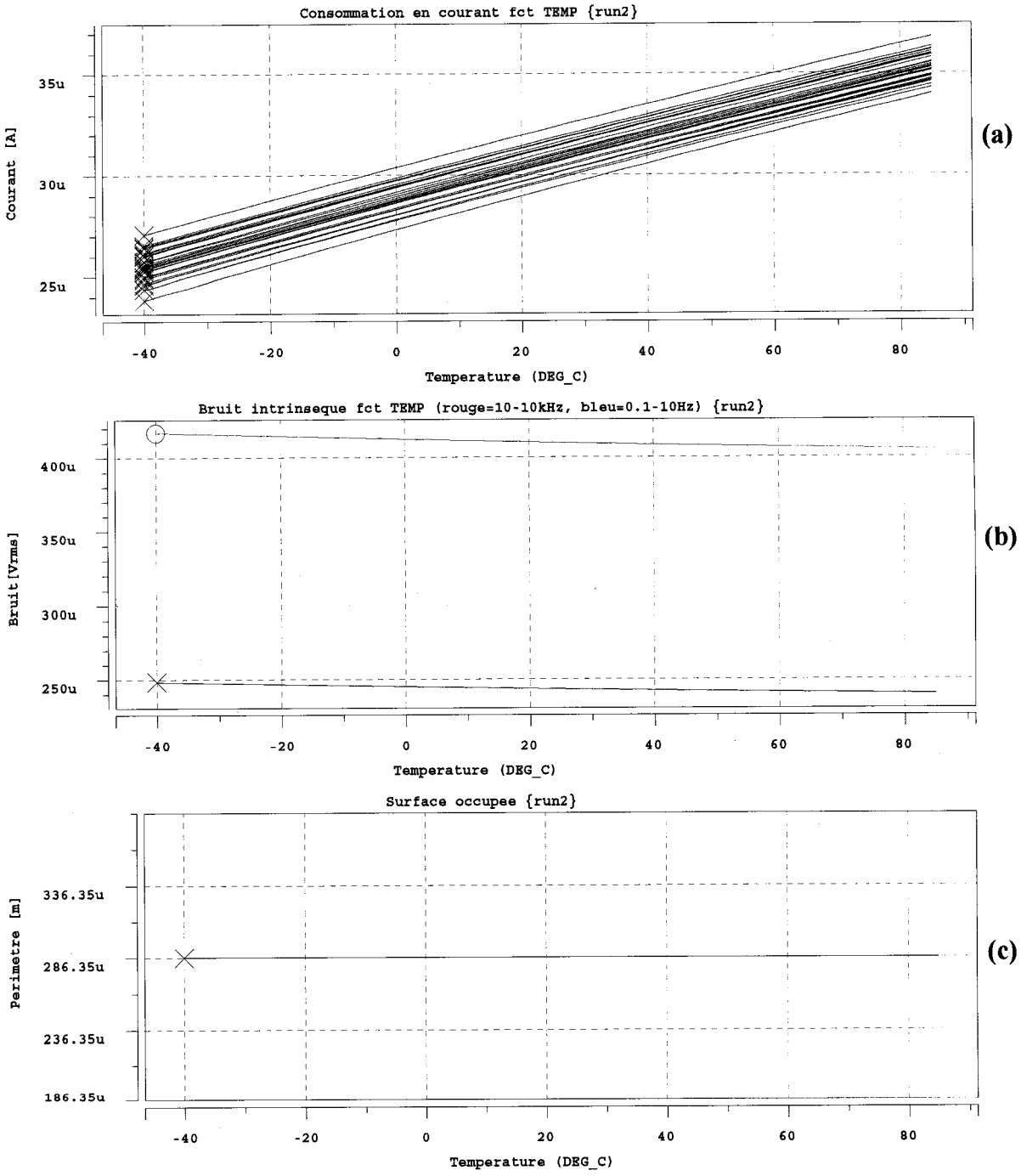


Figure 4.14 Courbes des performances de l'essai 2: (a) consommation en courant, (b) bruit intrinsèque, (c) surface occupée.

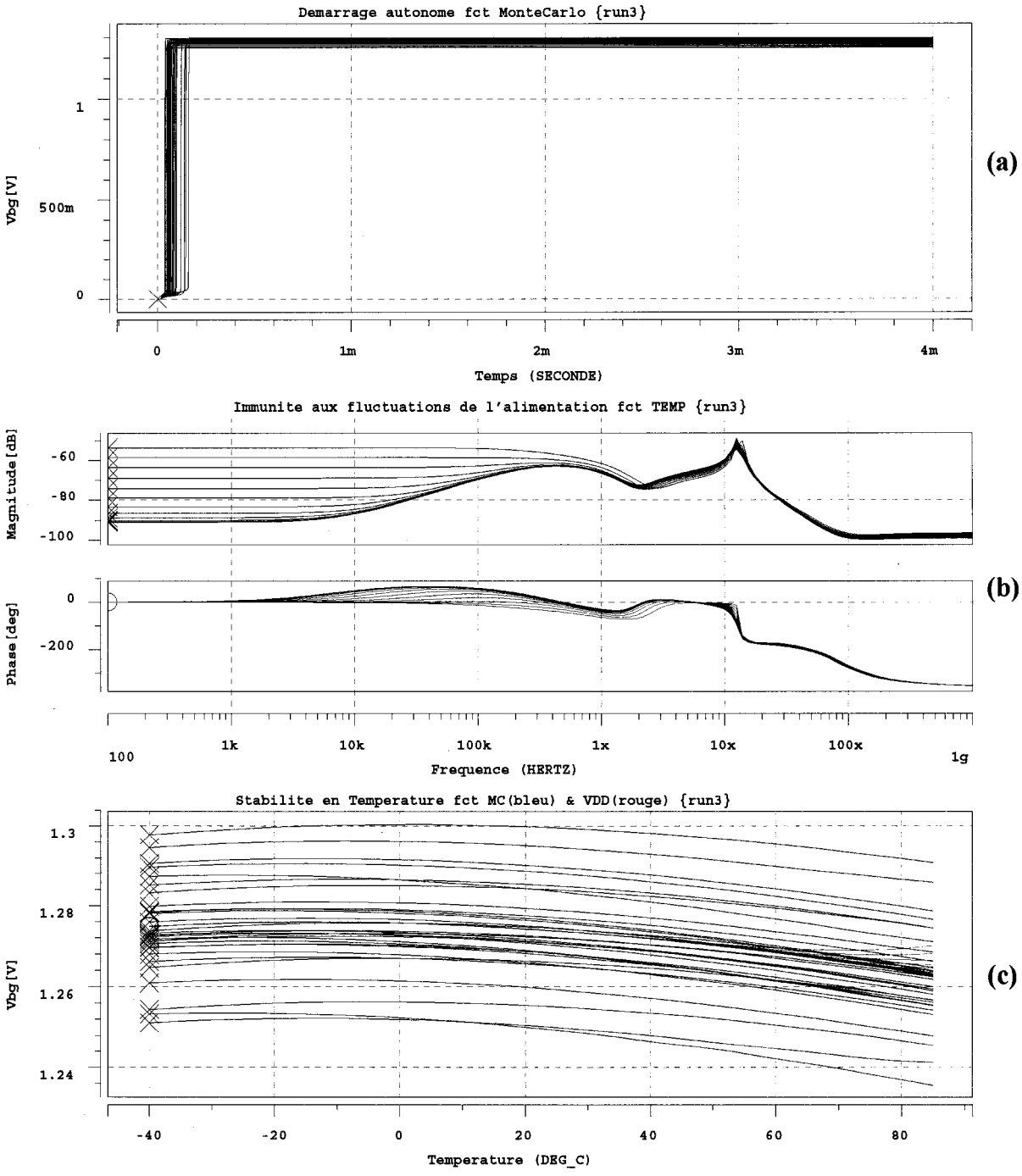


Figure 4.15 Courbes des performances de l'essai 3: (a) démarrage autonome, (b) immunité aux fluctuations de l'alimentation, (c) stabilité en température.

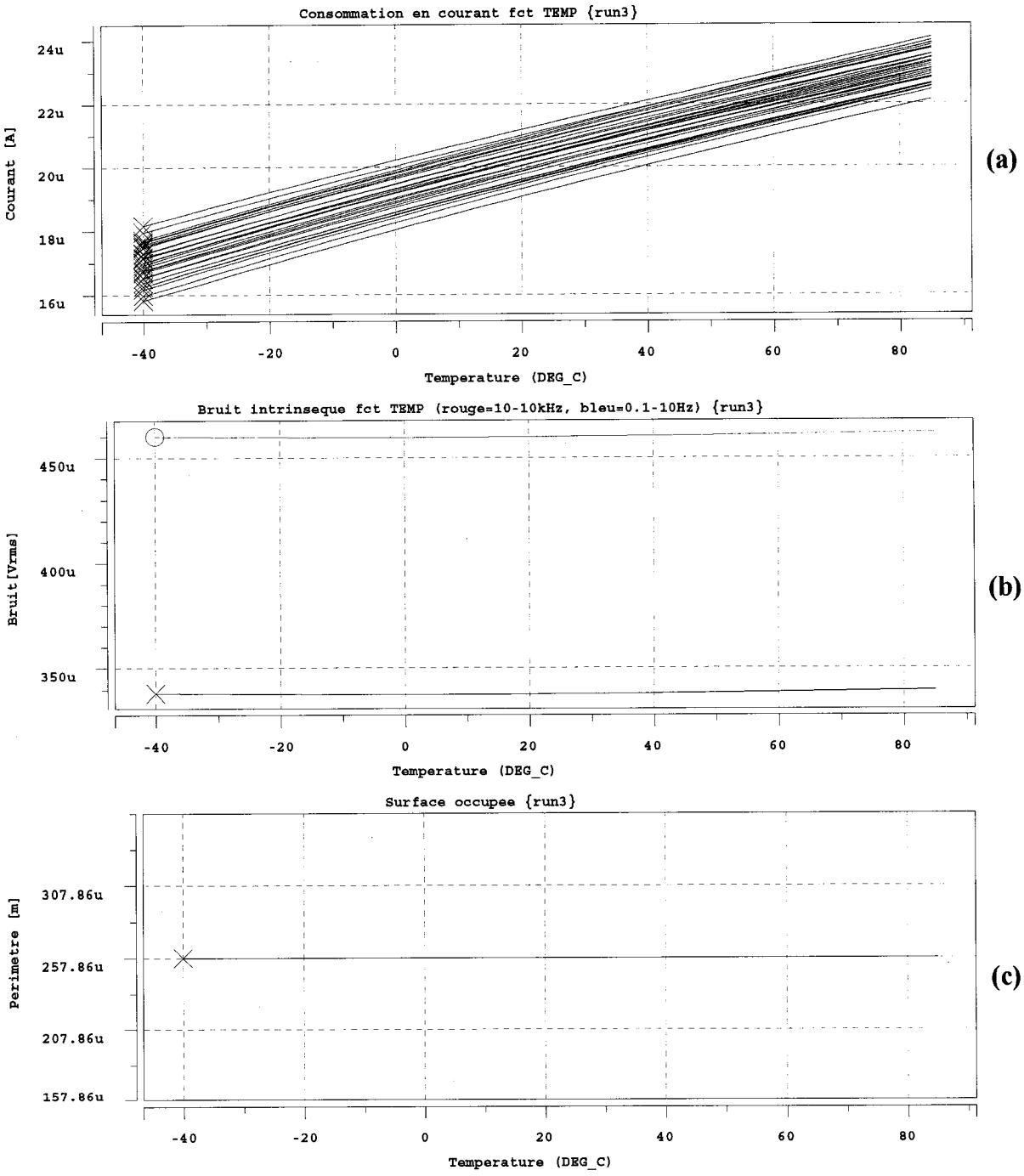


Figure 4.16 Courbes des performances de l'essai 3: (a) consommation en courant, (b) bruit intrinsèque, (c) surface occupée.

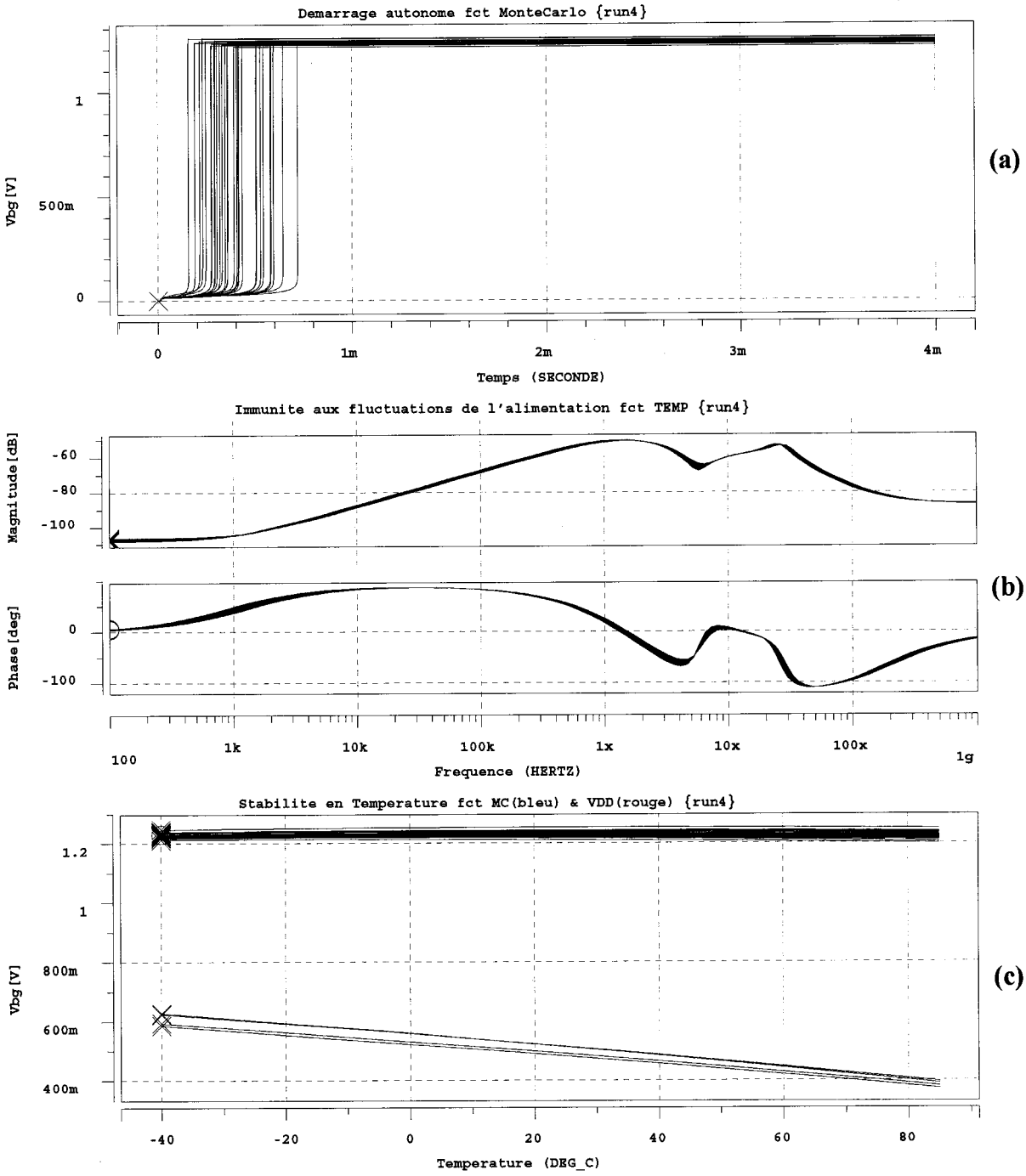


Figure 4.17 Courbes des performances de l'essai 4: (a) démarrage autonome, (b) immunité aux fluctuations de l'alimentation, (c) stabilité en température.

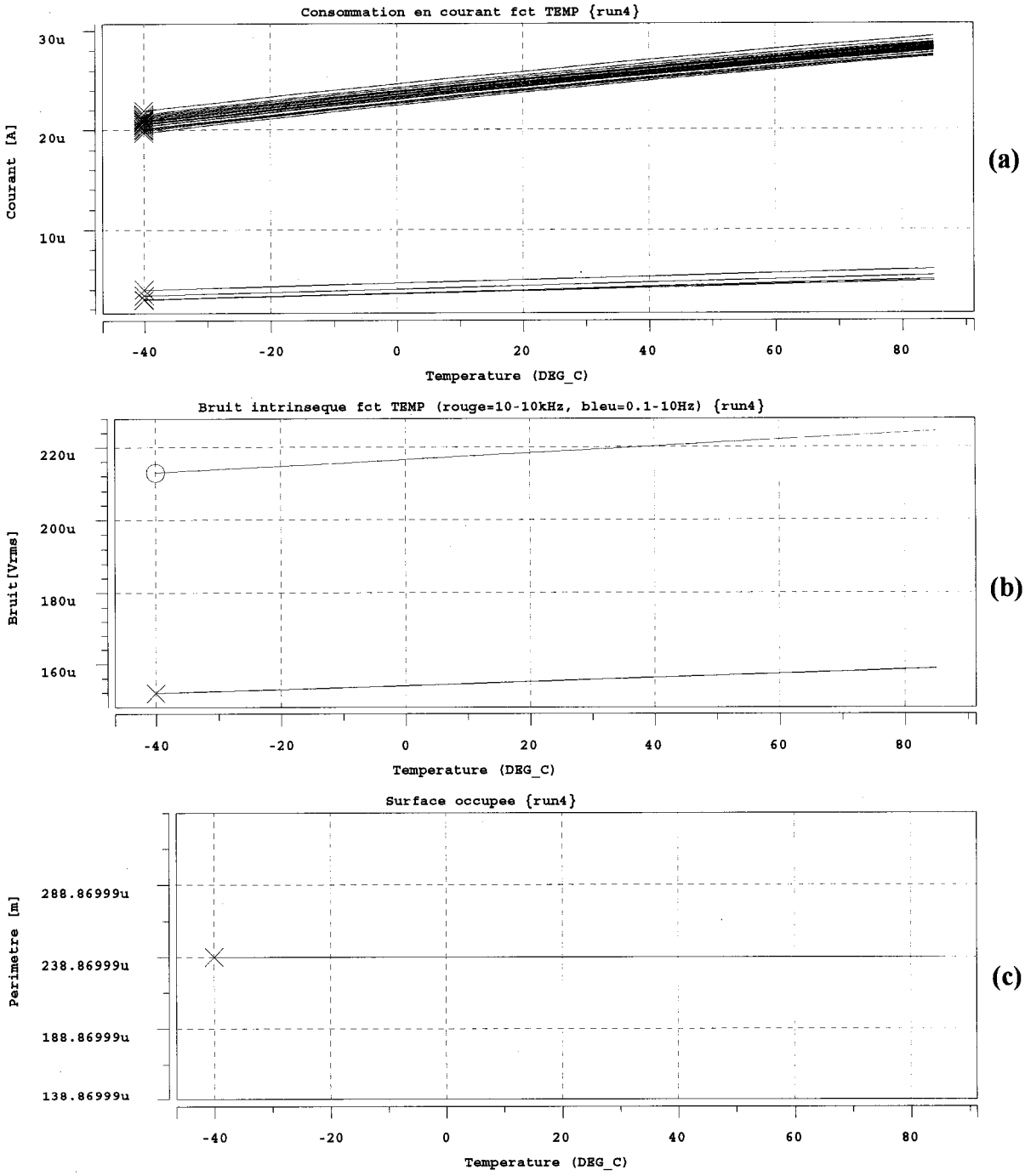


Figure 4.18 Courbes des performances de l'essai 4: (a) consommation en courant, (b) bruit intrinsèque, (c) surface occupée.

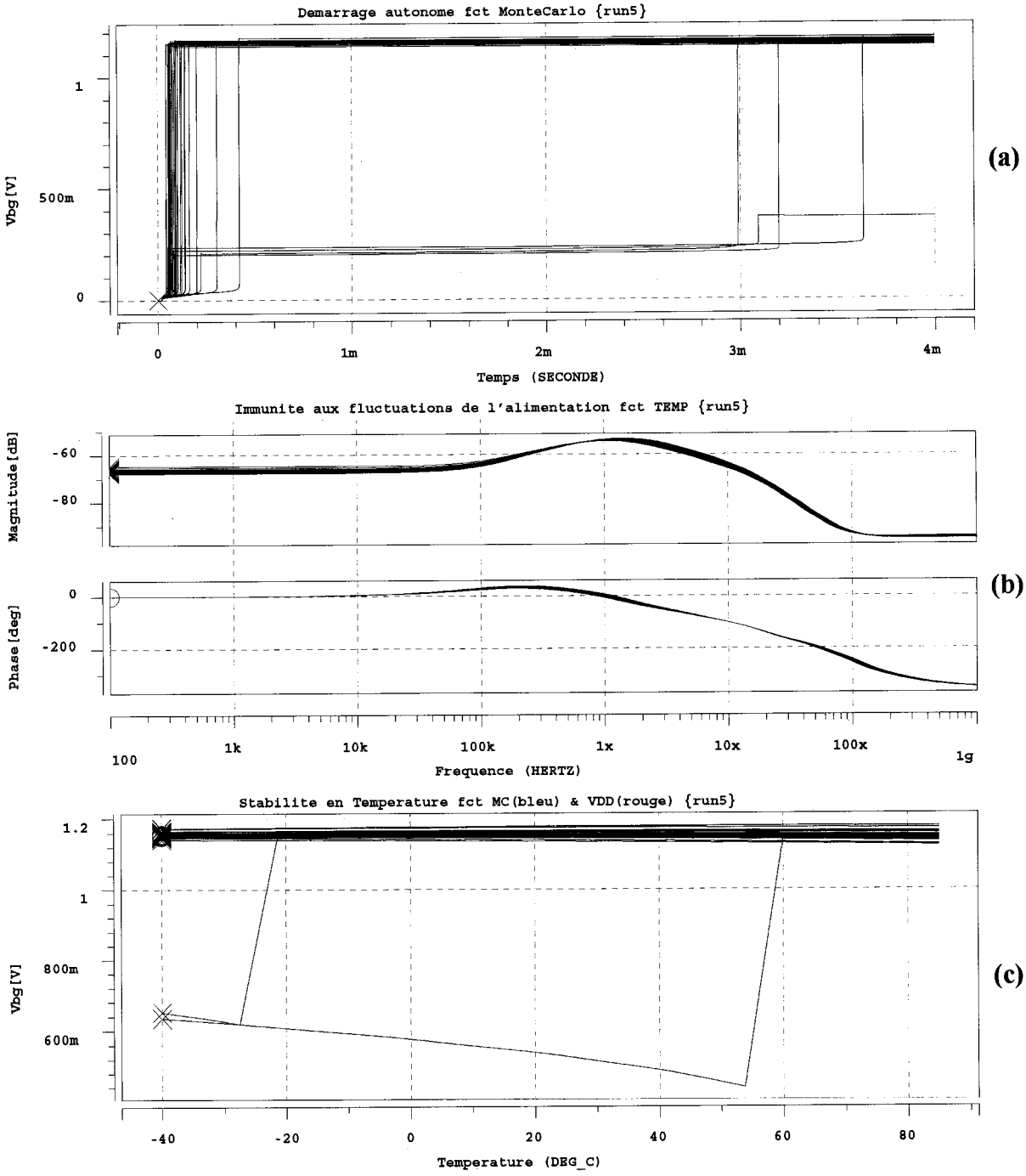


Figure 4.19 Courbes des performances de l'essai 5: (a) démarrage autonome, (b) immunité aux fluctuations de l'alimentation, (c) stabilité en température.

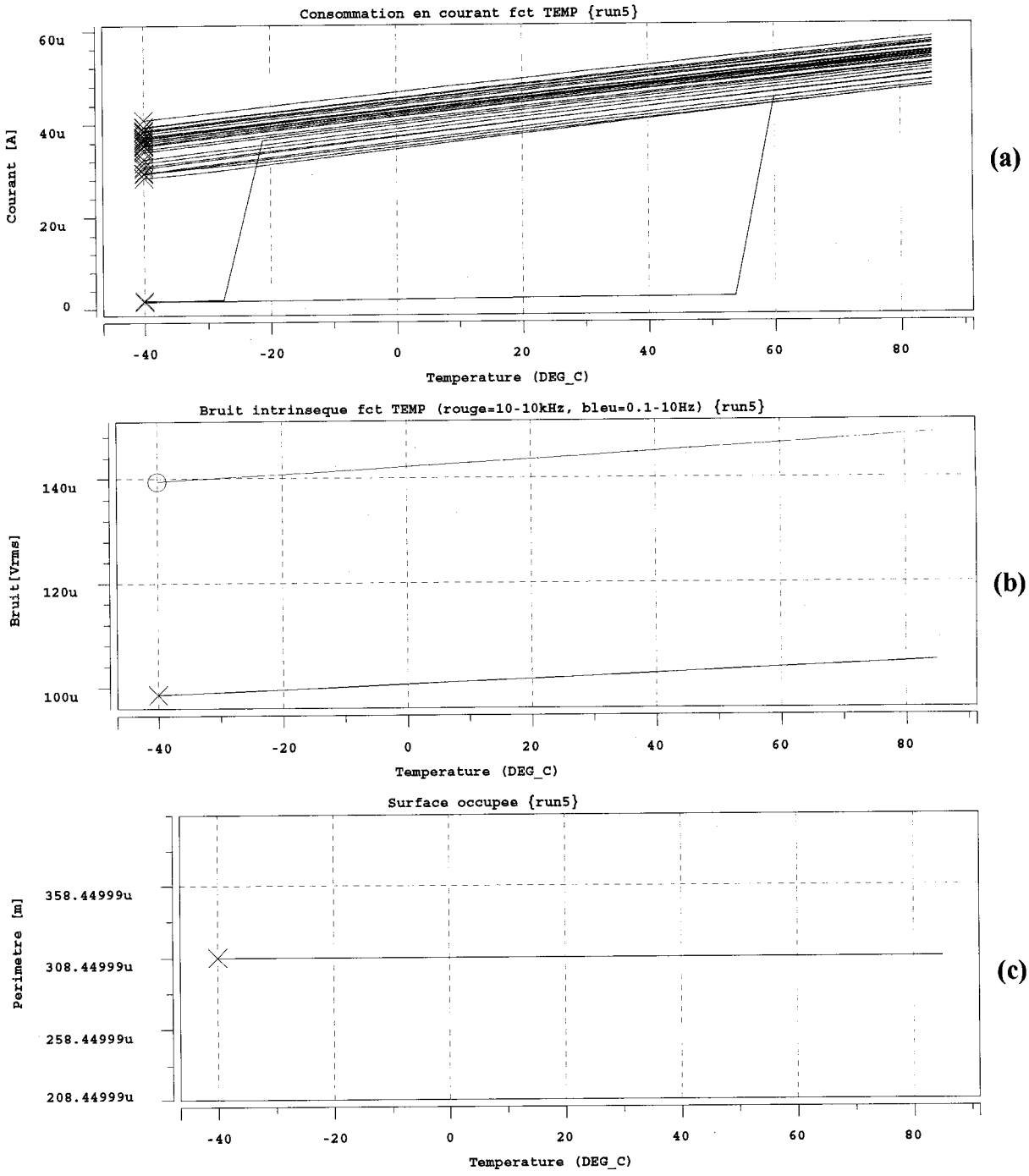


Figure 4.20 Courbes des performances de l'essai 5: (a) consommation en courant, (b) bruit intrinsèque, (c) surface occupée.

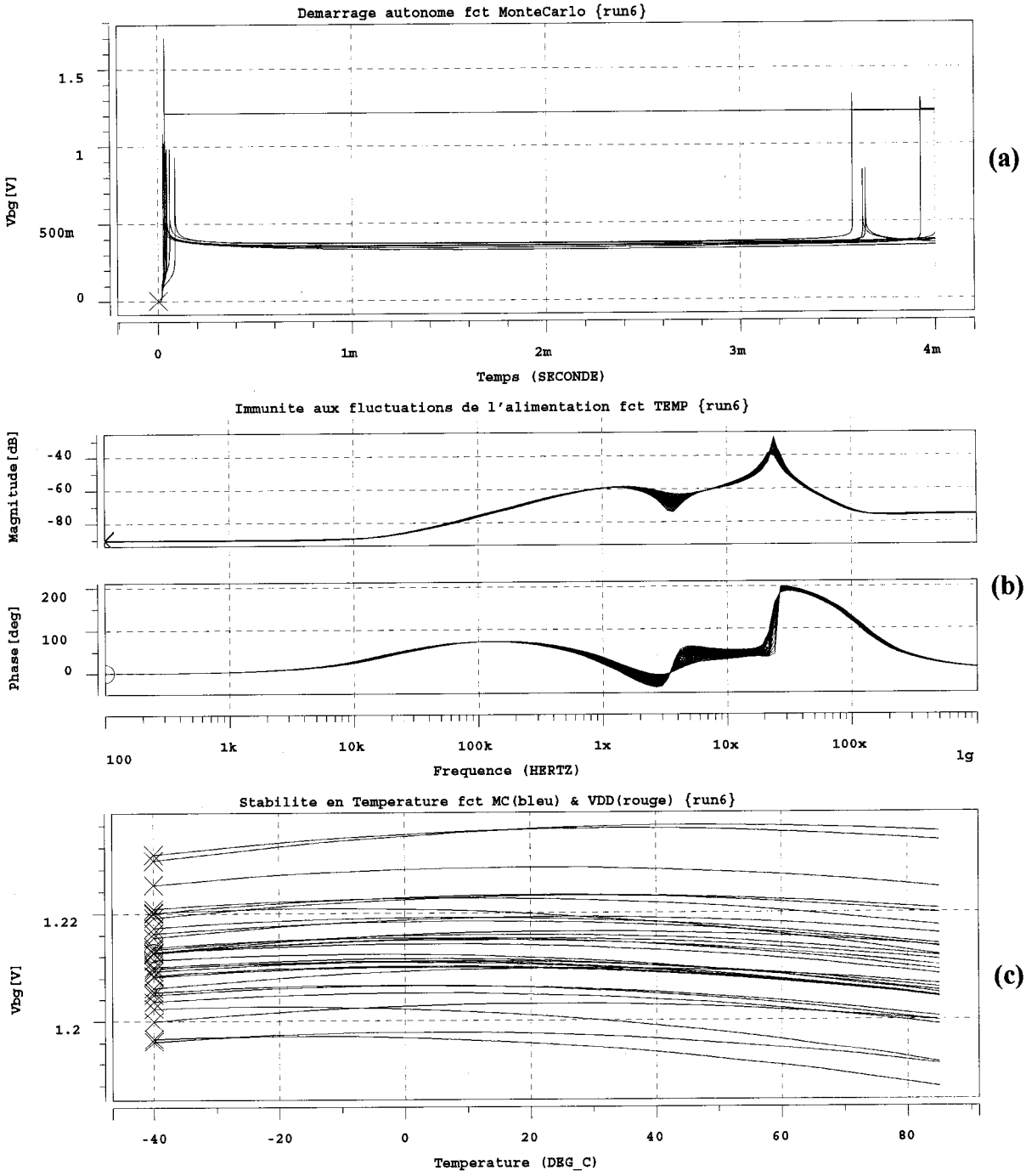


Figure 4.21 Courbes des performances de l'essai 6: (a) démarrage autonome, (b) immunité aux fluctuations de l'alimentation, (c) stabilité en température.

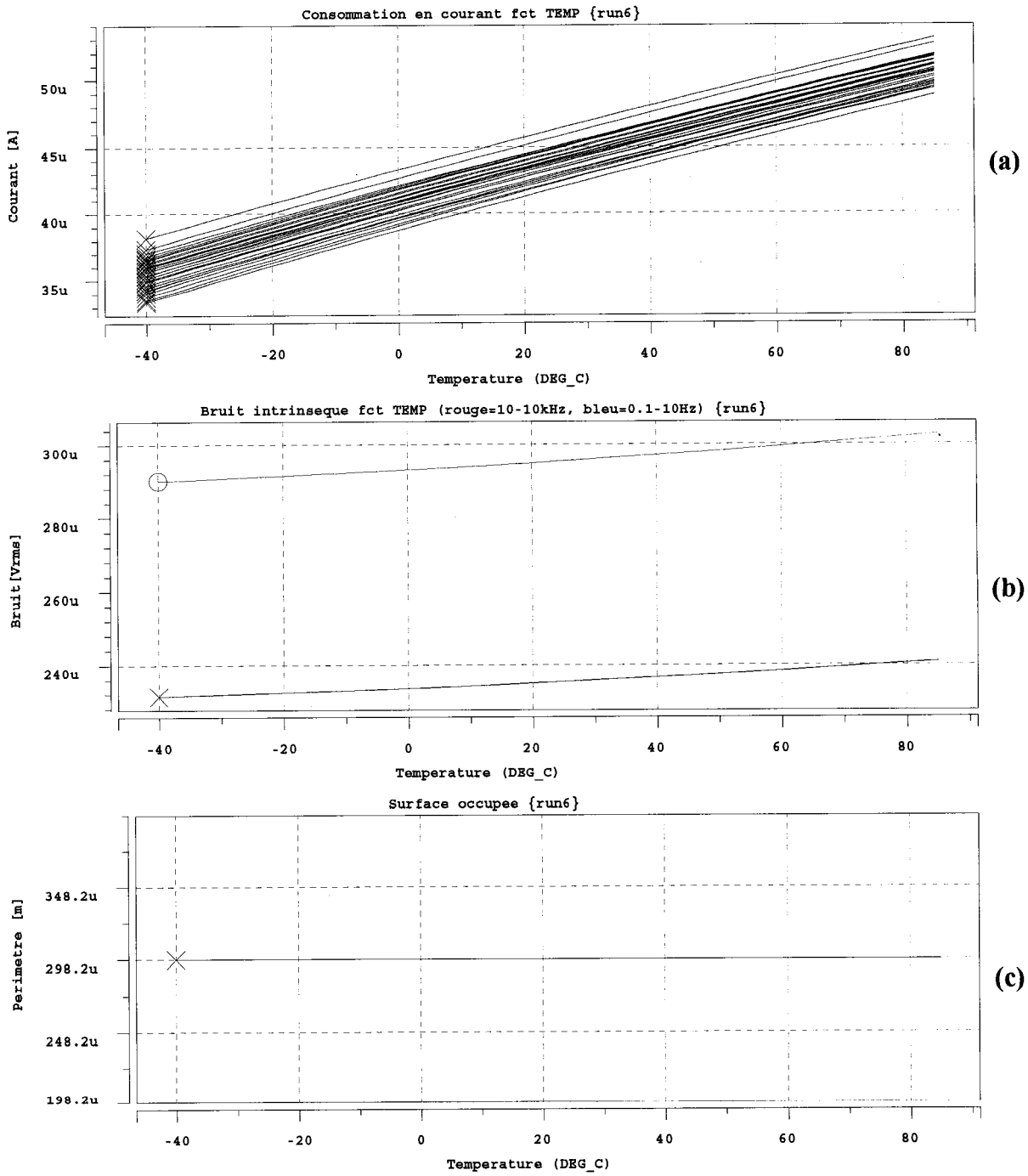


Figure 4.22 Courbes des performances de l'essai 6: (a) consommation en courant, (b) bruit intrinsèque, (c) surface occupée.

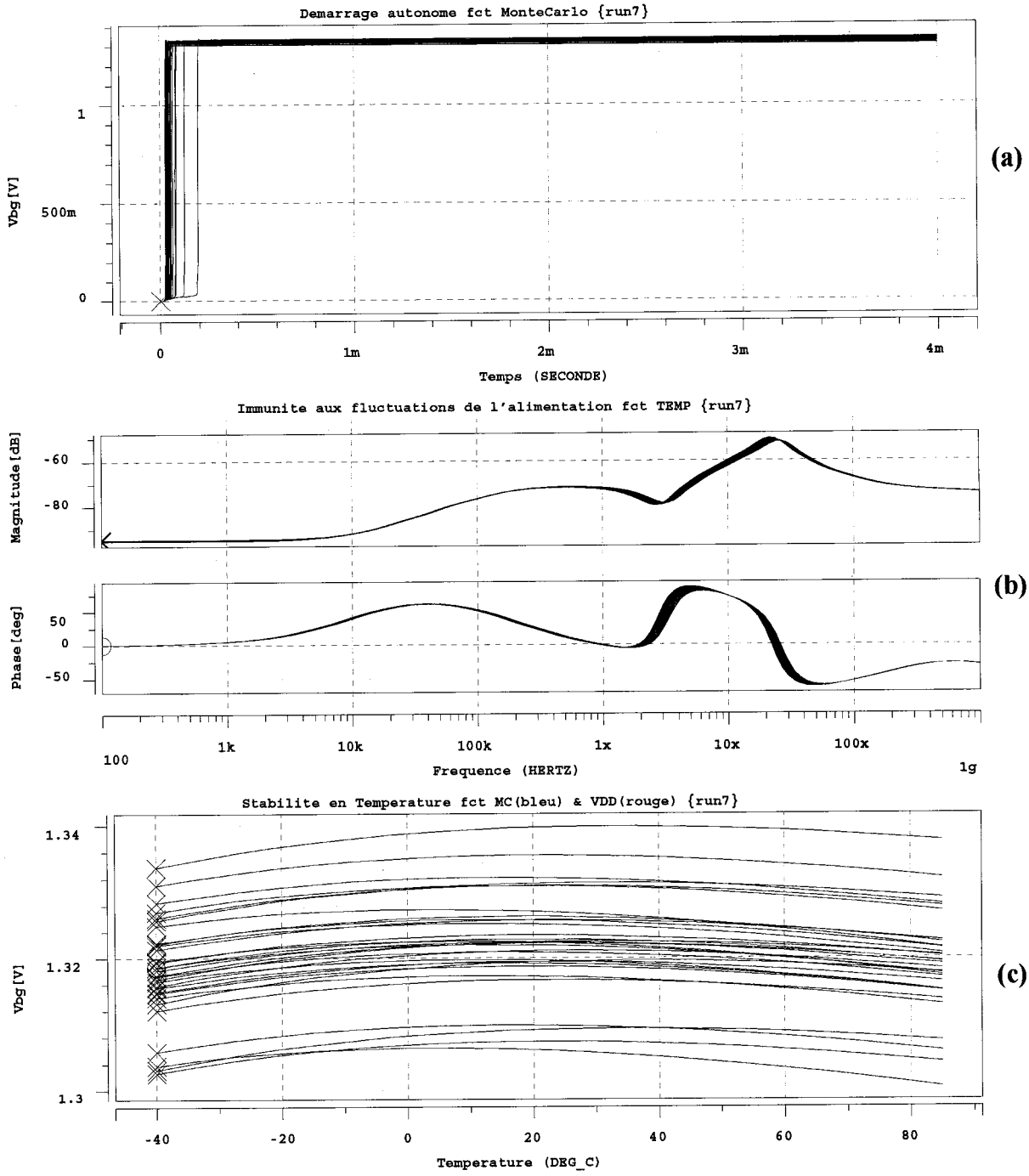


Figure 4.23 Courbes des performances de l'essai 7: (a) démarrage autonome, (b) immunité aux fluctuations de l'alimentation, (c) stabilité en température.

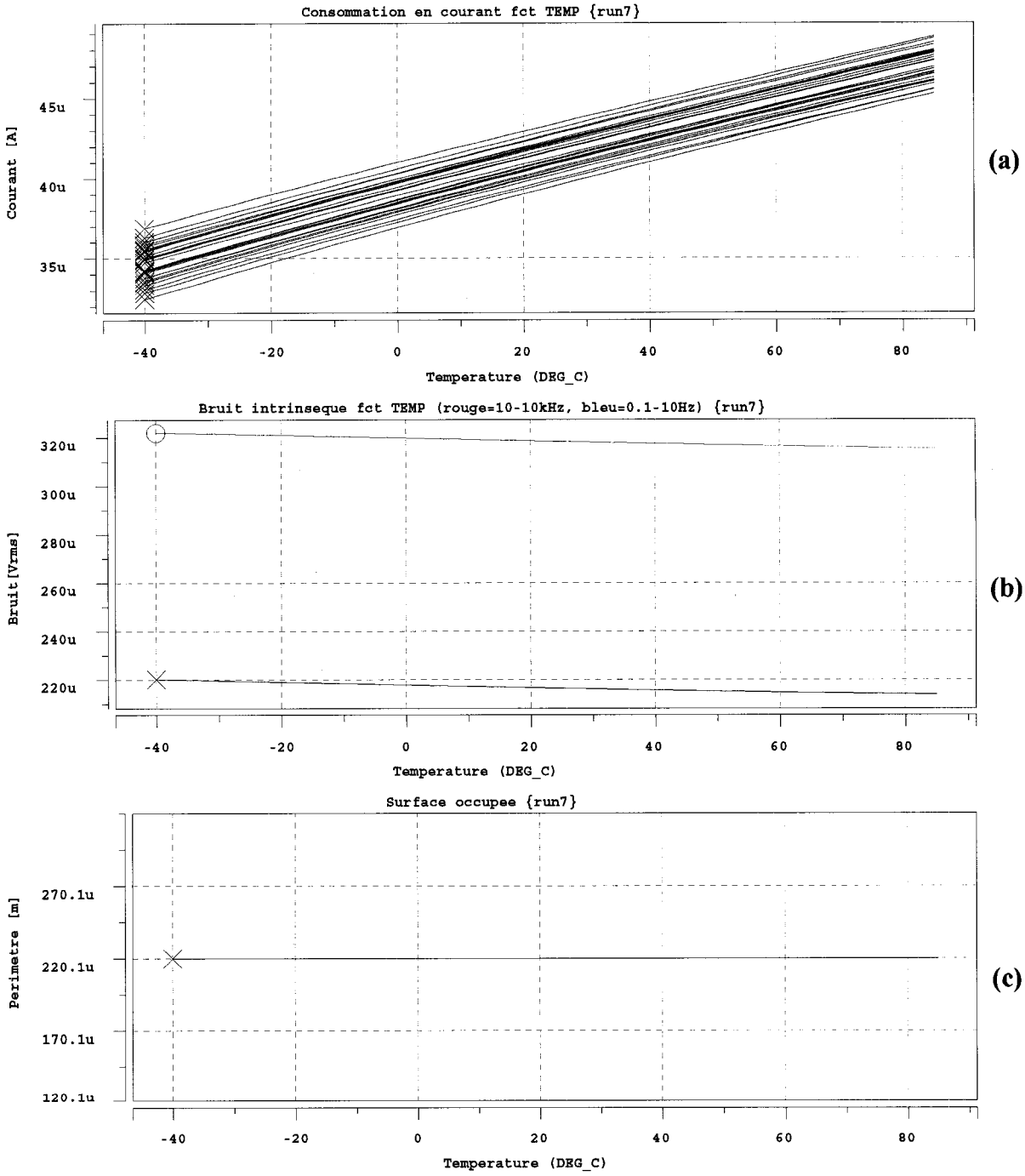


Figure 4.24 Courbes des performances de l'essai 7: (a) consommation en courant, (b) bruit intrinsèque, (c) surface occupée.

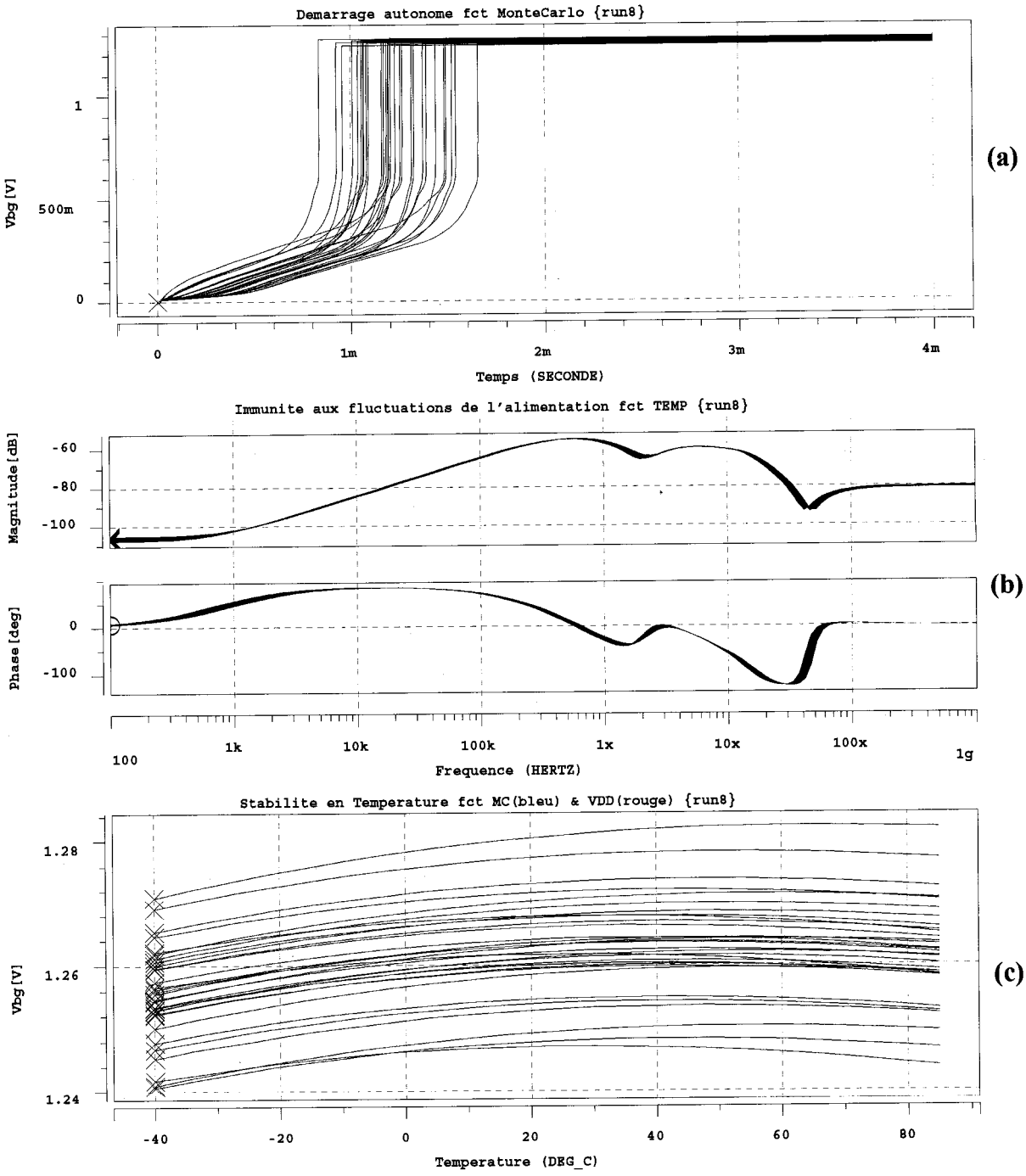


Figure 4.25 Courbes des performances de l'essai 8: (a) démarrage autonome, (b) immunité aux fluctuations de l'alimentation, (c) stabilité en température.

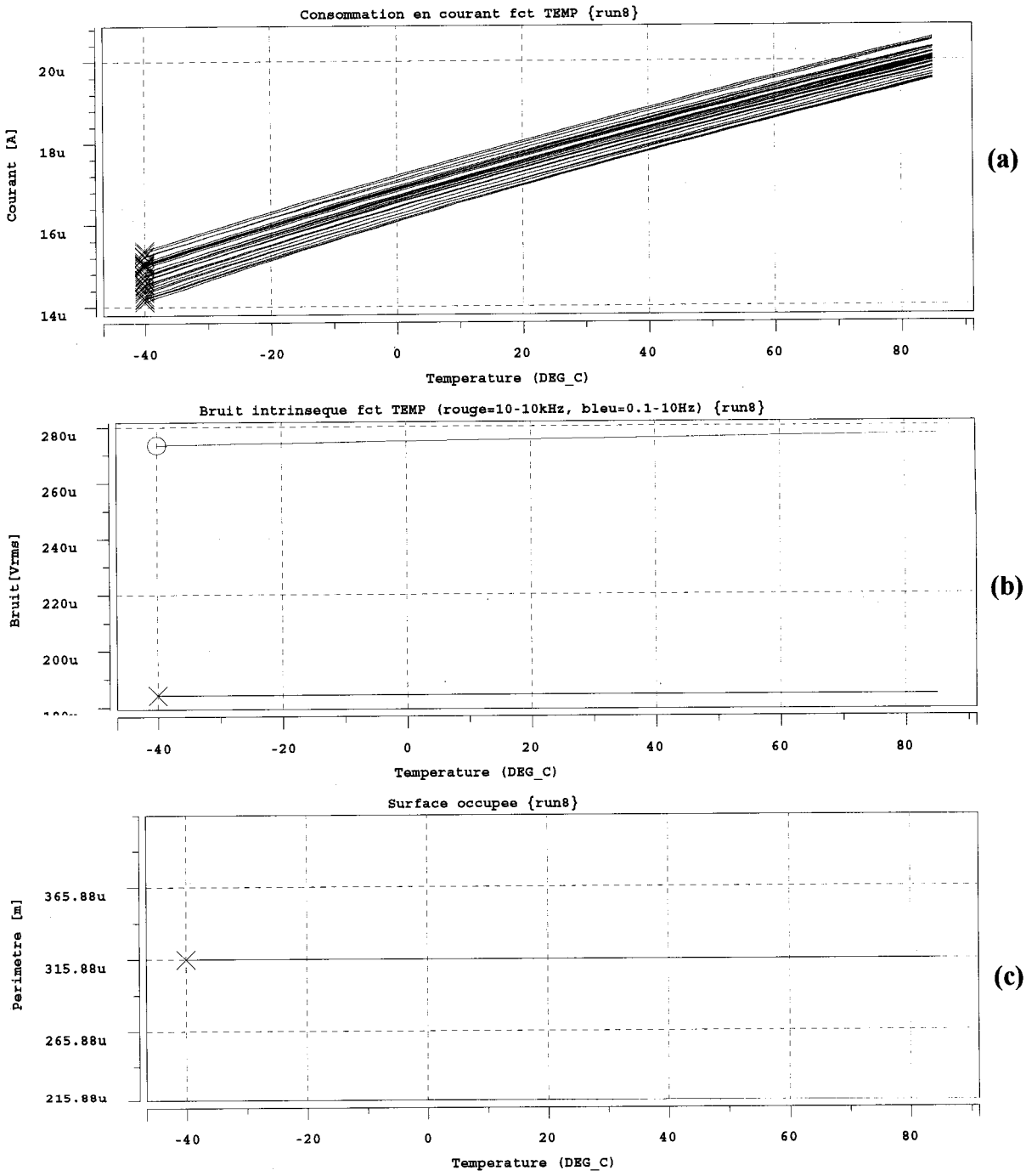


Figure 4.26 Courbes des performances de l'essai 8: (a) consommation en courant, (b) bruit intrinsèque, (c) surface occupée.

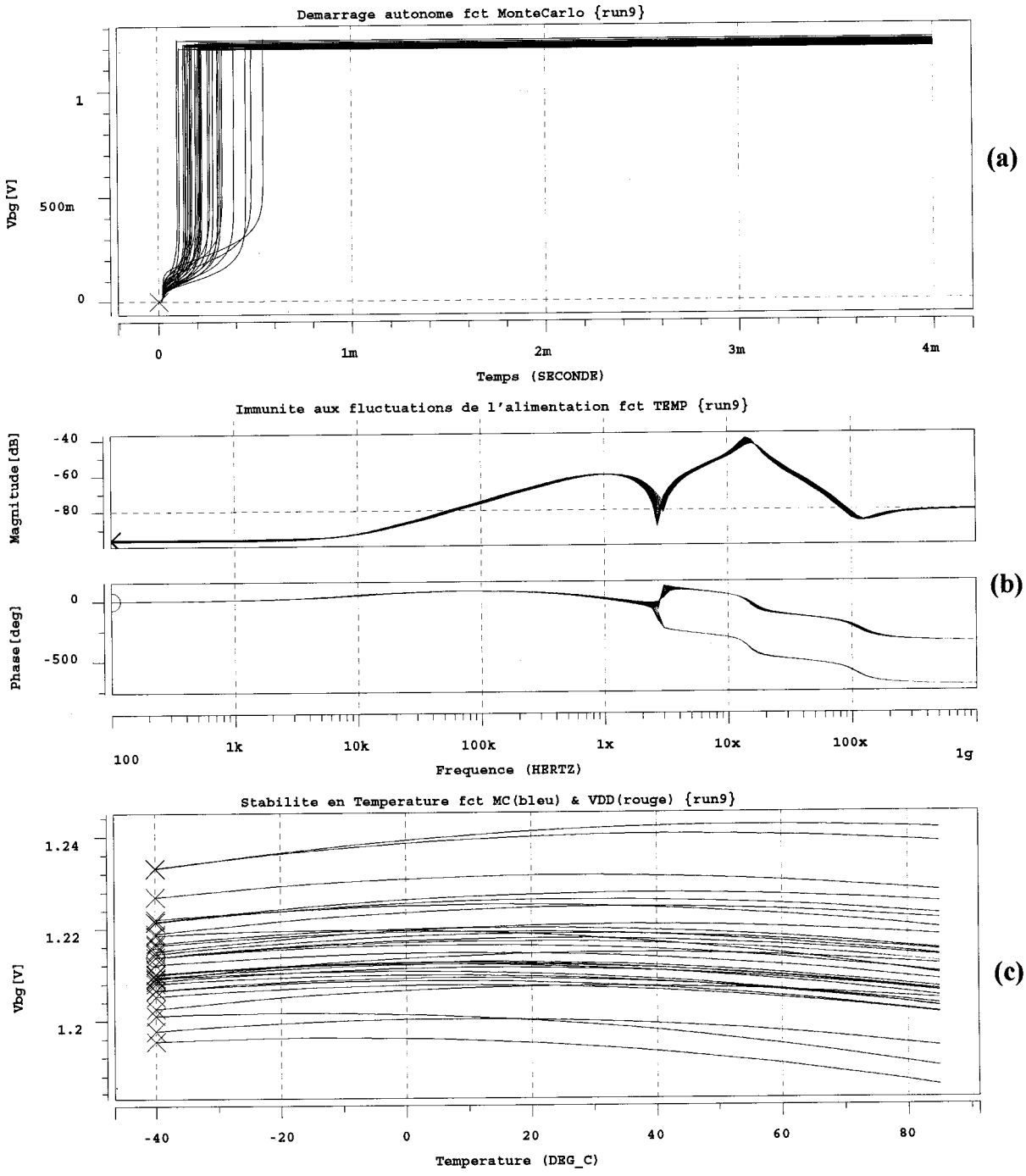


Figure 4.27 Courbes des performances de l'essai 9: (a) démarrage autonome, (b) immunité aux fluctuations de l'alimentation, (c) stabilité en température.

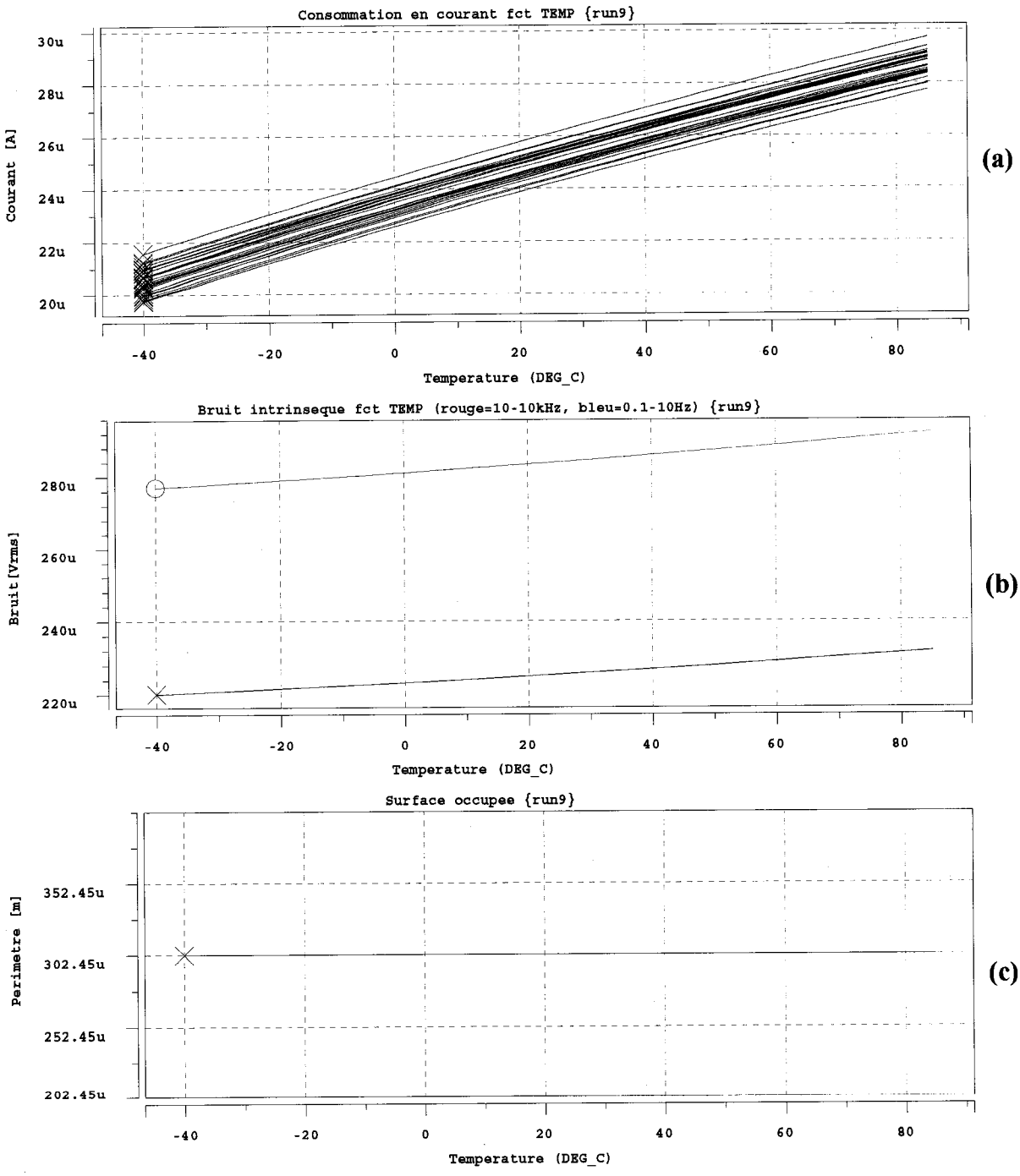


Figure 4.28 Courbes des performances de l'essai 9: (a) consommation en courant, (b) bruit intrinsèque, (c) surface occupée.

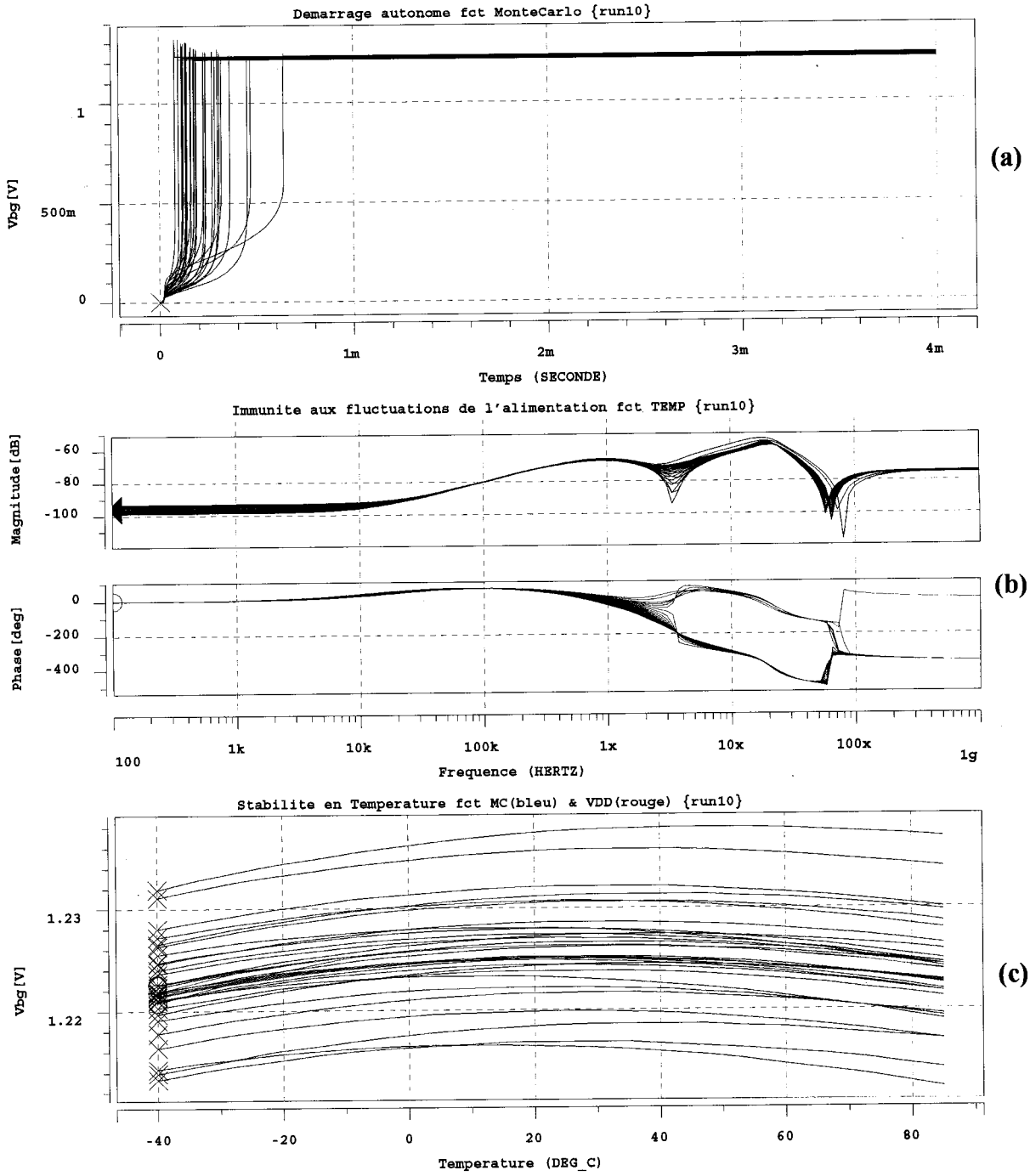


Figure 4.29 Courbes des performances de l'essai 10: (a) démarrage autonome, (b) immunité aux fluctuations de l'alimentation, (c) stabilité en température.

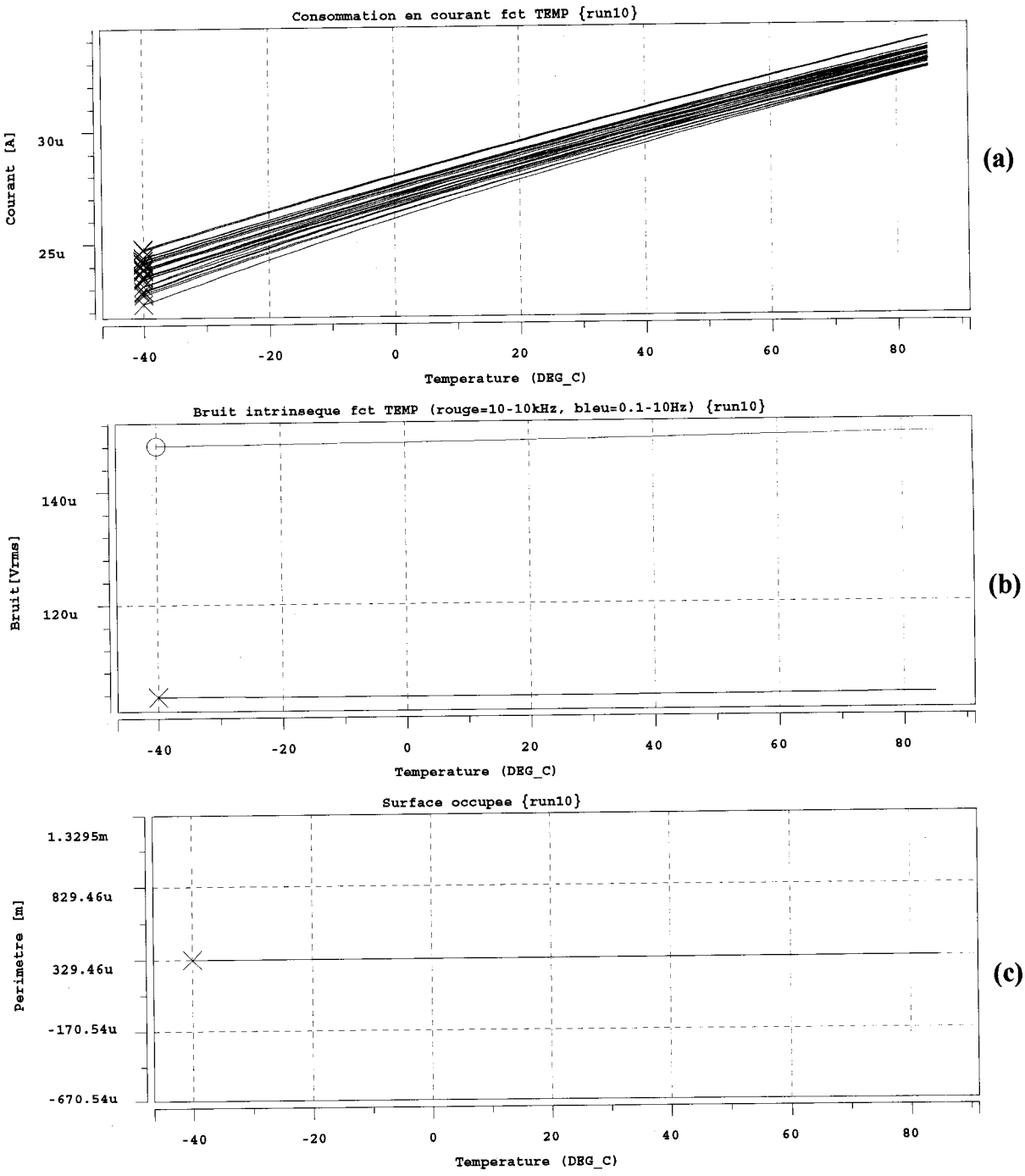


Figure 4.30 Courbes des performances de l'essai 10: (a) consommation en courant, (b) bruit intrinsèque, (c) surface occupée.